

NBS  
Publi-  
cations



# OMNIDATA

An Interactive System for Data Retrieval,  
Statistical and Graphical Analysis,  
and  
Data-Base Management

\*\*\*

A User's Manual

\*\*\*



United States Department of Commerce  
National Bureau of Standards  
Handbook 125







JAN 2 1979-ref

79097

QC1

.U51  
110125  
1978

# OMNIDATA

## An Interactive System for Data Retrieval, Statistical and Graphical Analysis, and Data-Base Management

\*\*\*

### A User's Manual

\*\*\*

by  
**Joseph Hilsenrath**  
and  
**Bettijoyce Breen**

Office of Standard Reference Data  
National Bureau of Standards  
Washington, D.C. 20234



U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary  
Dr. Sidney Harman, Under Secretary  
Jordan J. Baruch, Assistant Secretary for Science and Technology  
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

US  
'''

Issued September 1978

Library of Congress Catalog Card Number: 78-600076

**National Bureau of Standards Handbook 125**

Nat. Bur. Stand. (U.S.), Handb. 125, 294 pages (Sept. 1978)

CODEN: NBSHAP

U.S. GOVERNMENT PRINTING OFFICE  
WASHINGTON: 1978

## Preface

This handbook describes the characteristics, operation, and application of a general-purpose system for data retrieval, data analysis, and data file maintenance. The system has been designed so that persons with little or no knowledge of computers are able to search computerized data files and prepare ad hoc or periodic reports. Although designed with the novice in view, the system is of use to the computer professional as well. Numerous utility modules provide the data-base administrator with tools for maintaining the integrity of the data bases under his control. For the management staff, the system can provide answers—sometimes within minutes, often within the hour—to questions requiring computer processing of stored data.

The system consists of 45 unique modules and a main supervisory program. These carry on a dialogue with the user at a teletypewriter or video terminal to achieve a highly interactive system which operates as well in the *batch* mode as in *demand* mode on the NBS computer (UNIVAC<sup>®</sup> 1108) under EXEC 8.

The programs were designed and written by the authors, members of the Data Systems Design Group in the Office of Standard Reference Data, to meet the data handling and analysis needs of the diverse data analysis centers at NBS. Prior experience in the design of general-purpose programs pointed to the feasibility of a system which could handle not only the diverse scientific data files such as crystal data, thermochemical data, chemical kinetics data, and physical properties of substances and mixtures, but also bibliographic and administrative data bases as well. In the last category the system has been applied with equal success to the: personnel file; staff training file; equipment inventory file; carpool information file; project control file; foreign visitors file; foreign travel file; etc.

During the five or so years over which this program was developed and tested, the authors have received advice and encouragement from a number of colleagues at NBS and elsewhere. Among those who provided technical advice are Alfred Beam and Thomas Hall of Language and Systems Development, and Robert Thompson and Robert McClenon of the NBS Office of Standard Reference Data. Among those early users who coped cheerfully with Omnidata's growing pains are Betty Conrad of the NBS Personnel Division, Robert L. Getis of Auerbach Associates, and Roy Mullinax of the Department of Housing and Urban Development.

In the editorial sphere, special thanks are due: Constance L. Seymour, first for her patience in carrying out revision after revision of the manuscript and in the end for serving as the resident typist-typographer; Anne H. Meininger and Donald D. Wagman, for their painstaking reading of the manuscript for both technical content and editorial details; and finally to Carla Messina, whose innovative software allowed these words to flow from a teletypewriter terminal to a magnetic tape capable of driving a computerized typesetter.



## CONTENTS

1.0	Introduction.....	1
1.1	A Few Words on Words .....	4
1.2	Typical Files Handled Routinely by Omnidata.....	6
1.3	Typical Retrieval and Analysis Applications of Omnidata .....	7
1.4	Summaries of Modules of Interest to File Users.....	9
1.5	Summaries of Modules for File Building and Maintenance .....	13
1.6	Access to the Omnidata System .....	14
1.6.1	The Supervisory Program Omnidata .....	16
1.6.2	Global Instructions .....	16
1.6.3	System and File Security .....	21
1.6.4	Error Recovery .....	24
1.7	References.....	25
2.0	Conversion of Conventional Files into Omnidata Format .....	27
2.1	The DEFINE Program.....	27
2.2	File Definition when the Original Data is in File Format on Mass Storage .....	31
2.3	File Definition when the Original Data is on Cards or Card Images on Mass Storage.....	38
3.0	Searching, Reporting and Updating of Omnidata Files .....	42
3.1	The SEARCH Module .....	45
3.1.1	The EXIT and PRINT Options in the SEARCH Module .....	52
3.2	The DISPLAY Module .....	57
3.3	The REPORT Module .....	62
3.4	The UPDATE Module .....	72
3.5	Operating Omnidata in the Batch Mode and Remote Batch .....	77
4.0	An In-Depth Application of Omnidata to the NBS Crystal Data File .....	81
5.0	Descriptions of the Data Manipulation and Analysis Modules .....	103
5.01	ABRIDGE .....	104
5.02	AGGREGATE .....	107
5.03	ANALYSIS.....	113
5.04	ARRAY.....	117
5.05	BROWSE .....	121
5.06	COMPUTE .....	125
5.07	CONCAT .....	127
5.08	CROSTAB.....	129
5.09	DESCRIBE.....	138
5.10	DISPLAY .....	140
5.11	DISTRIBUTE.....	143
5.12	ENCODE.....	147
5.13	EXTRACT.....	153
5.14	FETCH.....	155
5.15	FIT.....	156
5.16	GRAPH.....	166
5.17	KWOC .....	171
5.18	PLAN.....	183
5.19	PLOT .....	187
5.20	RANDOM .....	199
5.21	REGRESS .....	201
5.22	RENAME.....	210
5.23	REPORT .....	213
5.24	SAVE .....	216
5.25	SEARCH .....	218
5.26	SEGMENT .....	223
5.27	SEQUENCE .....	225
5.28	SORT .....	226
5.29	STACK.....	229
5.30	STATIS .....	231
5.31	STATPLOTS .....	236
5.32	SUMMARY .....	244
5.33	SURVEY.....	246
5.34	TALLY .....	256
5.35	TRIM .....	265

6.0 Descriptions of the Utility Modules ..... 269

6.1 ANNEX.....	270	6.5 DICTIONARY.....	278
6.2 ATTACH.....	274	6.6 MOVE.....	279
6.3 BLANKS.....	276	6.7 SCREEN.....	281
6.4 CHECK SUM.....	277	6.8 USERS.....	285

## 1. Introduction

Judging from past experience, it is a rare manager, indeed, who is fortunate enough to obtain quickly or inexpensively answers to nonstandard questions on one or more aspects of the status of his organization. Computer programs designed to produce periodic reports are almost always inflexible, and the writing of ad hoc programs to answer specific questions entails expense and delays out of all proportion to the urgency of the problem that motivated the question in the first place. Efforts to simplify retrieval from computerized data files have produced, in recent years, numerous general-purpose programs. A number of the more successful commercial systems have already exceeded a few million dollars in sales. Surveys of commercial data base management systems appear from time to time in such magazines as *Datamation*, *Computer Decisions*, *Infosystems*, etc.

Most of the existing systems have adequate and roughly comparable search and arithmetic capability, file definition features, and more or less flexible report generators. None of these has nearly enough data analysis and data manipulation facilities for handling the numerical and alphanumeric data files in an active scientific data analysis center or in any large commercial endeavor.

We have drawn upon NBS experience in the design of general-purpose mathematical and statistical analysis programs [1-7]\* and experience over the years with a large variety of time-shared computer systems to design and program a modular interactive data analysis and retrieval system to operate on the UNIVAC® 1108 under EXEC 8. The system operates either from a terminal in DEMAND mode, from a deck of cards in the BATCH mode, or in the REMOTE BATCH mode.

Program modularity has long been a hallmark of truly efficient computer programming and systems design. Indeed, in most systems the modularity is not necessarily seen by the user. Our system is quite different in this respect. The Omnidata system is as modular to the user as it is to the computer. As an example, if we wished to carry out a physical inventory on a .25% random sample of equipment costing between \$20,000 and \$50,000 and have the results of the random selection arranged by building and room number, we would require the following specific and distinct operations:

\*Numbers in brackets denote references listed in section 1.7.

® UNIVAC is a registered trademark of the Sperry Rand Corporation.

- a) A SEARCH through the entire inventory file for items costing between \$20,000 and \$50,000;
- b) Selection on a RANDOM basis of 25% of the records satisfying the above criterion;
- c) A SORT of the resulting file on building and room location; and,
- d) A printed REPORT containing the desired information.

In the above, the words SEARCH, RANDOM, SORT, and REPORT are names of modules which the Omnidata user must call in sequence to achieve the desired solution. In each module the user is asked to supply the requisite particulars to achieve the result required. After each module has done its work, the user has an opportunity to check the results before going on to the next operation. Such interaction with the data file is facilitated by requiring the user to perform each operation separately, and Omnidata has a number of interesting and useful ways of assisting the user in looking at the data in the file.

A brief description of the function of each of the modules that comprise the Omnidata system is given in sections 1.4 and 1.5, while a detailed description can be found in sections 5 and 6.

The 35 modules which are discussed in section 5 provide facilities for: searching and reporting; plotting and other graphical data analysis; arithmetic operations in general, and statistical analysis in particular; file partitioning and subsequent sequential analysis on subfiles; keyword indexing of bibliographic files; flagging, coding, and decoding of data items; analysis of questionnaires and surveys, and a large variety of data management and validation routines of use to both the user of the data and the data base administrator (the file builder).

The system is designed for quick response to complicated searches and analysis. The following is an example of a moderately complex operation that was performed easily by the system. Management wished to have, as soon as possible, a breakdown of the average age and average grade within each of approximately 30 occupational series in each major organizational unit separated into supervisory and nonsupervisory staff. How this problem is solved via the SORT module is explained later.

The following are examples of less complicated chores that can be performed by Omnidata within the hour—provided, of course, that the requisite data have been collected and are available either on punched cards, magnetic tape, or mass storage devices.

- a) A cross tabulation of age group versus salary levels.
- b) A frequency distribution of staff by levels of education.
- c) The geographic distribution by ZIP codes of subscribers.
- d) A complete statistical analysis of purchases in excess of \$5000 for a specified period of time.
- e) A classification of respondents into ten age groups (deciles).
- f) A tally of universities attended by senior staff.
- g) A list of references of papers treating alloys of zinc and copper.
- h) A report of missing data items in a file.
- i) A cross tabulation of age groups (20, 25, 30, 35, 40) versus professions of all blood donors of age 40 years or less.

- j) A random 10% sample from the inventory file of equipment costing between \$5,000 and \$20,000 ordered by building and room location, and showing the date of purchase.
- k) A list of book titles indexed under tables and thermodynamics, but excluding steam tables.
- l) A listing of report numbers of classified reports carrying unclassified titles, listed by corporate author in chronological order of report date.
- m) A distribution of active research grants in excess of \$50,000/year by states in decreasing order of total expenditures since initiation, or of total expenditures in excess of \$200,000 regardless of current level of support.
- n) A listing of inorganic cubic crystals containing phosphorous arranged by name in alphabetic order.
- o) A complete statistical analysis of staff salaries for each major organizational unit.

An important feature of the Omnidata system is its ability to generate data arrays so as to be accessible to other programs (written in FORTRAN, COBOL, XBASIC, etc.) which do not have the retrieval facilities of Omnidata.

Another important feature of Omnidata is its ability to interface with the OMNITAB II program [7] which is used extensively at NBS and elsewhere. This interface gives OMNITAB II users an easy to use, yet sophisticated, search facility while providing the Omnidata user with a repertory of well-tested and highly accurate statistical programs [8-11].

The Omnidata system has been designed to serve as a facile tool for nontechnical users (clerks, secretaries, administrators, etc.) as well as for the data-base manager (computer programmer, analyst, file builder). While the former group is concerned mainly with framing questions (file searching) and getting answers in the form of reports, the latter group is, in addition, concerned with file definition, file updating, data editing and other manipulative facilities.

The seven modules of interest to the latter type of user have been grouped together under the heading of "Utility Modules." These are described briefly in section 1.5 and in more detail in section 6. For the general user, the system provides 35 distinct modules for a wide variety of data manipulations and analysis in addition to the searching and reporting facilities found in other management information systems. These modules are described briefly in section 1.4 and more fully in section 5.

In section 2 we describe DEFINE, an independent XBASIC program, which takes conventional data files and converts them to Omnidata format. In section 3 we discuss in detail four of the Omnidata modules. Three of these (SEARCH, DISPLAY, REPORT) allow one to address questions to data files, look at final or intermediate results, and obtain plain or fancy reports. The fourth is the UPDATE module which allows the data file administrator to make corrections to individual records or to all records in a systematic fashion. These together with the DEFINE module represent a minimum of modules with which the user must be familiar in order to be able to retrieve information from a computerized data file.

## 1.1 A Few Words on Words

Aside from the vocabulary of normal discourse, this manual makes use of words having quite specific connotations. These words can be divided into four main classes. The first of these contains words which are names of Omnidata modules. Among these are DEFINE, ABRIDGE, STACK, SUMMARY, ANALYSIS, COMPUTE, RENAME, etc. Each of these modules performs a number of quite specific operations which are discussed briefly in sections 1.4 and 1.5 and more fully in sections 3, 4, and 5.

The second class of words contain general global Omnidata instructions like TERSE, TIME, WIDTH, MONITOR, etc., which control how the system as a whole operates. These are discussed in section 1.6 and again when necessary in the detailed description of the modules in sections 3, 4, and 5.

The third class consists of words such as current, previous, exit, restart, select, reject, end, etc. These have meanings specific to certain of the Omnidata modules and are discussed as needed in sections 3, 4, and 5.

In the fourth class are a few general words needed to discuss data storage, retrieval, and analysis, such as data items, data vectors, physical records, logical records, data files, etc. It seems important to clarify some of these here.

- a) A *data item* or a *data entry* is an individual datum, such as a person's age or salary, or year of birth, which is numeric; part numbers which may be a mixture of letters and numbers; or a name, a country, a city, a job title, etc., which is usually alphabetic.
- b) A *logical record* is made up of all of the data items pertaining to a particular person or incident or case. Thus, all of the data values associated with the attributes name, social security number, date of birth, grade, salary, job title, etc., make up a logical record in a personnel file.

The use of the modifier *logical* serves to distinguish this collection of information from a *physical record* which refers to the data portion of a magnetic tape as in tape record or to a machine record, which need not concern the normal user of the Omnidata system. As the Omnidata user will normally not be concerned with tape records or machine records, we have sometimes used the word *record* without its modifier to signify a logical record.

- c) A *data vector* as used here is a list of numbers or character strings made up of all of the data entries (one from each logical record) for a single attribute. If all of the data items for a single person are called a logical record, then all of the ages (one from each record) constitute the *age vector* in the file. In a data file consisting of the information shown in tabular form, the rows would be logical records and the columns would be data vectors.

In a data file of short logical records, a tape record or a computer record would consist of many logical records. Conversely, a long logical record may extend over two or more physical records (either on tape or in mass storage).

- d) A *data field* is the space (number of characters) allotted in each record for a particular data item.
- e) An Omnidata *data file* consists of the totality of logical records, where each record consists of the same number of data items arranged in the same order (with due allowance for missing data items). This file can be resident either on mass storage (disc) or on magnetic tape from which it can be "rolled in" to disc as required.

There are other words (like blocked records) which have general technical meaning and even some (like Fieldata) which have a special meaning—being peculiar to UNIVAC® computers. We consider it a service to the nonspecialist reader of this manual not to define these words. Computer specialists who know what these words mean will normally be involved in advising on or in generating the conventional files on which the Omnidata system will work. Their help will be useful or essential, as the case may be, but only up to the point of defining the file. After a file has been defined into the Omnidata format, no specialized training is required to search it or to analyze it.

\* \* \* N O T E S \* \* \*

## 1.2 Typical Files Handled Routinely by Omnidata

The system has been applied to files of quite varied format and data content. Some files that have been handled successfully are listed below.

- a) A personnel file containing 108 data items comprising 720 characters for each of approximately 4000 people.
- b) A chemical kinetics data file containing 17 data items for each of 500 chemical reactions of 484 interest in atmospheric chemistry and air pollution.
- c) A carpooling data file containing 35 data items for 3546 persons.
- d) A data file containing 17 data items for each of 83 foreign research projects.
- e) A file of selected values of chemical thermodynamic properties containing 18 data items for each of the 9563 records.
- f) An index of state building codes.
- g) Design characteristics and sales data for minicomputers.
- h) A file of cost data on computer runs containing 10 data items for each of approximately 4000 runs.
- i) A file of data on 1126 fires containing 110 data items extending over 1962 characters per fire.
- j) A project summary file containing 108 data items for approximately 100 projects.
- k) A file of records of committee and subcommittee memberships containing 26 data items for each of 1500 persons.
- l) Twelve thousand card images, each of which contains responses to 46 questions in a survey of users of stations WWV and WWVH.
- m) A series of 12 coordinated and consonant files (having the same record length and data layout) comprising a total of 24,000 records containing the NBS Crystal Data System (300 characters per record).
- n) An historical file of personnel actions (accessions, promotions, reassignments, and separations) over a 10-year period.
- o) A cumulative record of training courses taken by individual staff members.

The above enumerated files were originally in fixed-field form or have been converted to fixed field by a utility program.

A few words are in order concerning the seemingly large number of data items in certain of these files, as well as the smallness of others. If Omnidata were simply a data retrieval system, we would discourage its use on files as small as some in the above list. It is the numerous data analysis and display features which make it profitable to automate files as small as some on the list.

Most of the available management information systems (that use inverted file structures) limit the number of data items that can be searched. To avoid the problems inherent in building and updating an inverted file system [16], the Omnidata system operates on its files in a sequential manner. Besides, an inverted file system would speed up the operation of only 2 or 3 of the 40 odd modules in Omnidata. Consequently every data item in a file can be searched in Omnidata—even down to the character level if desired.

### 1.3 Typical Retrieval and Analysis Applications of Omnidata

In the previous section, we listed a variety of files which are handled more or less routinely by Omnidata. We shall now describe a variety of problems these files are able to solve via the Omnidata modules.

**Problem:** Select from a file persons whose: educational level is 17 or greater; academic discipline is in the range 0501 through 0517; and whose duty station is any one of four locations (designated by nine digit numbers). Sort results on organizational unit, then by educational level, then by discipline, then by duty station. Finally, list names in alphabetic order, and add age to the above items in the report.

**Remarks:** The solution of this problem involved the modules SEARCH, SORT, COMPUTE, and REPORT.

**Problem:** Extract from a chemical kinetics data file those records in which the reactants are H and H<sub>2</sub>O, and the products include H<sub>2</sub>O<sub>2</sub>. Print the reactions, the reference, the coefficients (A, B, C), and their uncertainties.

**Remarks:** The solution to this problem involved the modules SEARCH and DISPLAY.

**Problem:** Extract from the main carpool file all persons who wish to utilize bus transportation, and prepare a report showing the population density as a function of grid coordinates.

**Remarks:** The solution to this problem involved SEARCH, followed by the use of the EXTRACT module (to get the two high-order digits from the five digit X and Y coordinates). Next, the CONCAT operation generates a four-digit field equivalent to a two mile by two mile square. The population in each of these squares was produced by using the TALLY module on the concatenated field.

**Problem:** Prepare a series of summary tables and frequency distributions showing country by country the following: total number of projects; total funding to date (by intervals of \$10,000); distribution among scientific disciplines; and a set of cross tabulations showing sponsoring division against project, country, and discipline.

**Remarks:** The solution of this problem involved the modules SORT, TALLY, ENCODE, DISTRIBUTE, SUMMARIZE, and CROSSTAB.

**Problem:** Search the chemical thermodynamics data file for boron—containing molecules in the gaseous state whose entropy is between 80 and 90, and whose specific heat is less than 25, cal/deg mol.

**Remarks:** This problem required only the SEARCH and DISPLAY modules.

Problem: List by states in alphabetic order all existing codes (building, electrical, and plumbing) covering public buildings, but excluding schools and colleges.

Remarks: This problem involved use of the SEARCH (in the SELECT and REJECT modes), SORT, and REPORT modules.

Problem: Total and display computer costs for 10 publications in spite of the fact that different project designations were used on the individual runs for the same publications.

Remarks: This problem required the use of the modules AGGREGATE, SEARCH, SUMMARIZE, and DISPLAY.

Problem: Prepare a monthly summary listing all foreign visitors and showing their distribution among countries, host organizational units, sponsoring agencies, length of stay, and scientific disciplines.

Remarks: This periodic report makes use of the PLAN module which calls the modules SORT, DISTRIBUTE, TALLY, and REPORT.

Problem: Prepare a semiannual summary showing the involvement of staff members in committees, subcommittees, and working groups of professional societies. Show distribution by societies and divisions of societies, and by organizational unit, by type of appointment, and by length of service.

Remarks: This report involves the modules AGGREGATE and TALLY.

\* \* \* N O T E S \* \* \*

## 1.4 Summaries of Modules of Interest to File Users

A brief summary is now in order outlining the important implemented features of the Omnidata system for: information retrieval; numerical, statistical, and graphical data analysis; data display and report generation; data screening and correction; file definition and updating; and file manipulation. Where examples are given below, they are only for illustrative purposes and not necessarily the most important use of the module.

Module Name	Description
ABRIDGE	Permits the extraction or exclusion of certain data vectors to produce a smaller file of the more active data items. (See also SEGMENT.)
AGGREGATE	Allows for classifying records into broader classes on the basis of specific entries in any of the data vectors. Thus, persons with a dozen or more varied job titles can be classified as engineers. In the degenerate case, this module can be used to translate entries in the file on a one-to-one basis.
ANALYSIS	Displays the total, minimum value, maximum value, the average, and the standard deviation of the mean for numeric data in specified data fields. (See also SUMMARY.)
ARRAY	Extracts all of the numeric data vectors from a file and stores them on disc in a two-dimensional array suitably dimensioned and formatted to be accessible to programs outside of the Omnidata system.
BROWSE	Allows one to browse through the file on any one data field at a time to see how the information is stored. (See also DISPLAY.)
COMPUTE	Permits arithmetic operation on numerical data vectors with the view of augmenting the original file with data vectors resulting from such operations on existing vectors. The new data vectors can be used in subsequent operations by other modules during the run. They can also be retained permanently by saving the augmented file.
CONCAT	Provides for concatenation of two or more data items into a single vector; to provide, for example, a sort key or shorthand for a complex search performed periodically.
CROSSTAB	Generates a cross tabulation of data vectors, two at a time (e.g., pay vs. age).

DESCRIBE	Provides the user with an on-line description of each of the data items in the current file (provided that the file manager has generated such a description via the DICTIONARY module.)
DISPLAY	Provides a quick look at any or all of the data items in specified records in the current file. Each data item is suitably labeled. (See also BROWSE.)
DISTRIBUTE	Breaks a file into catalogued subfiles on the basis of the entries in a particular data vector for subsequent independent or tandem analysis, or other operations.
ENCODE	Groups numerical data in a designated vector into specified class intervals for subsequent analysis, display, or cross tabulation.
EXTRACT	Extracts from a defined data field certain contiguous characters and assigns a label to them so that the fragment itself becomes a new data element. The year, month, and day can be extracted from a data field to be concatenated later in the order suitable for sorting.
FETCH	Brings into the ambit of Omnidata a previously catalogued Omnidata file not otherwise available in the current run.
FIT	Interfaces with the OMNITAB II system to perform a least-squares fit of a polynomial of specified degree to one data vector as a function of another.
GRAPH	Produces bar graphs with considerable variation in format.
KWOC	Prepares a keyword (out of context) index of the words or fragments in a designated data vector.
PLAN	Permits Omnidata to get its instructions from a catalogued file, instead of the keyboard to facilitate preparation of frequent or periodic reports.
PLOT	Interfaces with the OMNITAB II program to produce a plot of up to five data vectors as ordinate against one data vector as abscissa.
RANDOM	Generates a set of random numbers and randomly selects a designated portion of a file for auditing or other purposes.
REGRESS	Interfaces with the OMNITAB II program to perform a multiple linear regression on as many as 10 data vectors as a function of a specified data vector.
RENAME	Allows the user to change the names (labels) of the data vectors in an Omnidata file. These changes can be either ad hoc for reporting purposes or permanent.

REPORT	Provides for either ad hoc or periodic report generation with considerable flexibility for titles, footnotes, columnar spacing, and multilevel column headings.
SAVE	Allows for storing and cataloguing of files produced by certain of the Omnidata modules.
SEARCH	Allows for sophisticated data retrieval from Omnidata files, including: boolean logic, anchored or unanchored partial string searches (prefixes, suffixes, roots, stems, etc.), exact or relational matches, matches within numerical ranges, ignoring of specific intermediate characters, etc.
SEGMENT	Breaks a large file into segments containing specified numbers of records (either a fixed number or a variable number of records per segment).
SEQUENCE	Adds to an existing file a set of sequence numbers in ascending order with specified initial values and increment. It can also insert a specified character string to serve as a flag.
SORT	Provides options for sorting a file on the basis of information in as many as 10 data vectors. Options are available to strike totals and subtotals or local and global averages at as many as ten hierarchical levels.
STACK	Combines the designated consonant (having the same record length and the same data layout) Omnidata subfiles into a single composite file wherever it is advantageous or necessary for subsequent operations (sorting, searching, analysis, etc.).
STATIS	Does a statistical analysis, one vector at a time, giving a decile distribution and 34 measures of location, dispersion, randomness, and other statistical parameters.
STATPLOT	Provides four plots useful in exploratory data analysis as an adjunct to the numerical results obtained from the STATIS module. Like STATIS it operates on a single vector of data. This module interfaces with the OMNITAB II.
SUMMARY	Computes for each of the numeric data vectors in a file the total, minimum value, maximum value, the average, and the standard deviation of the mean. Prints out the above information in tabular form, adds the information to the data file, and sets a flag to document the operation. (See also ANALYSIS.)
SURVEY	Analyzes a file of coded information representing answers to multiple choice questions or results of extensive surveys. All of the responses are tallied separately for each question or item in the survey, reported in an array, and displayed in histogram form, if desired.

- TALLY** Produces a histogram showing the frequency of occurrence in a particular data vector (either numeric or alphabetic) of each unique entry. Cumulative frequency, percentage, and cumulative percentages are also printed when the format allows it.
- TRIM** Searches in each record of a file for trailing blanks in a designated data vector, and reports a census of character lengths found in that data vector. On the basis of this analysis the pointers can be modified in the label table to achieve operational economy and space compression in DISPLAY and REPORT.

**\* \* \* N O T E S \* \* \***

## 1.5 Summaries of Modules for File Building and Maintenance

- ANNEX** Permits on-line file building in either prompting or direct mode.
- ATTACH** Combines the corresponding records of two or more Omnidata files into a single file. The files must be in the same sorted order and contain an equal number of records, in one-to-one correspondence.
- BLANKS** Counts and reports, for each data field in the file, the number of records in which that data field is completely blank.
- CHECK SUM** Calculates two check sums at the time the file is generated. These are used to check the integrity of the file prior to and after periodic or ad hoc updates.
- DICTIONARY** This module builds a file of information required by the DESCRIBE and the SURVEY modules. That file will describe the content and format of the data vectors associated with a particular data base. When the data file contains encoded information the file will contain information decoding it.
- MOVE** Provides for moving records or blocks of records from one place in a file to another.
- SCREEN** Searches in a designated data field (through all of the records in the file) and reports the number of occurrences of each unique character in each character position allotted to that data field.
- USERS** Allows the file builder or manager to enter into an Omnidata file (or delete from the file) the names and passwords for accredited users of that file, thereby giving him control of the security of the system, file by file.

## 1.6 Access to the Omnidata System

Omnidata is a self-contained package of programs written entirely in the XBASIC language as defined in the manual titled "XBASIC for UNIVAC® 1100 Series Computers" by Language and Systems Development, Inc. of Silver Spring, Maryland. The system consists of a main program, Omnidata, and the above enumerated modules which it supervises.

After the computer has accepted the normal UNIVAC® RUN control statement, as is evidenced by the date and time printout, the Omnidata system is called for in either of two ways depending upon the manner in which the system has been implemented. The most efficient manner in which to operate the Omnidata system is as a set of compiled absolute programs (modules). In this case it is necessary to call for it by typing

```
@NBS*OMNIDATA.OMNIDATA
```

For those agencies that do not have a compiler which can operate absolute code, the Omnidata system is run from XBASIC source programs which are compiled as needed. Under such circumstances the system would be called for by typing

```
@SBASIC,P OMNIDATA
```

if the run is in the DEMAND mode, or by typing

```
@SBASIC,PO OMNIDATA
```

if the run is to be made in BATCH.

An Omnidata run can be terminated by typing the word STOP at any time the program asks for an input. On receipt of such an input, the current run is terminated by the XBASIC compiler or by the absolute code, as the case may be. This action is accompanied by the printout:

```
PROGRAM STOPPED  
TIME: X.XXXX
```

At this point Omnidata is no longer in control, but the BASIC compiler still is.

In order to stop the entire run properly it is necessary to exit from both the XBASIC compiler and the system. Typing @EOF releases the XBASIC compiler and @FIN instructs the Executive operating system to close out the run and report the run charges, etc., as shown in figure 1.6a.

```

* * * * *
* >@xbasic
* XBASIC R5.1 15:29:07 8 MAR 77
* >old:omnidata*
* READY (A)
* >run
* OMNIDATA 15:29:17 8 MAR 77
* *PLEASE ENTER ACCOUNT NUMBER --->? >abc
* *TYPE PASSWORD --->? >xyz
* *WHICH DATA BASE DO YOU WANT --->? >fpr75
*
* GOOD AFTERNOON, WELCOME TO OMNIDATA.
* ** NOTE — OMNIDATA KEEPS A RECORD OF WHO USED WHICH
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* PROGRAM STOPPED. (B)
*
* TIME : 2.629 (C)
* >@eof
* LSD XBASIC 15:30:26 8MAR 77
* >@fin (D)
*
* RUNID: BREENB ACCT: 12345-ABCDEF PROJECT: BASICEXT
* TIME: TOTAL: 00:00:21.768 CBS: 00001195.877
* CPU: 00:00:02.849 I/O: 00:00:00.586
* CC/ER: 00:00:18.332 WAIT: 00:00:47.421
*
* CPU:$ 0.17 CBS:$ 2.15 I/O:$ 0.04 CC/
* TAPES: $ 0.00
* PRIORITY: DEMAND
* SURCHARGES: RUN: $0.25
* DEMAND ELAPSED: $ 0.09
* *TERMINAL INACTIVE*
* >@@term (E)
* * * * *

```

Figure 1.6a. Shown here is a record of the start and the proper termination of an Omnidata run. The asterisk (\*) at [A] indicates that the program has been made public to be available to all users. At [B] we exit from Omnidata. At [C] an input of @EOF releases the XBASIC compiler, and at [D] the input of @FIN terminates the run and produces the Executive accounting information. The input at [E] is required to free the communication port.

The UNIVAC® operating system allows for interrupting the program during a printout. The requirement to interrupt a printout will arise, if at all, in generating a lengthy report. The REPORT module has been programmed to permit such interrupts. How this is achieved is shown in section 3.3.

*1.6.1 The Supervisory Program—Omnidata.* The following are the major functions of the main program Omnidata:

- a) It obtains the user's identification and password and the name of the desired data file.
- b) It finds the designated file on main storage or, on failing to do so, prints out an appropriate diagnostic message. See appendix A for a list of possible diagnostics produced by the main Omnidata program.
- c) It reads the appropriate portion of the data file, and checks the user's name and password against the list of accredited users as stored in the particular file selected.
- d) It reads from the header records and stores a label table in core, consisting of the names and locations of each of the defined data elements in the record, while taking proper regard for the user's status on restricted data items.
- e) It obtains information on the length of the file and reports to the user the number of records and the number of data items per record.
- f) It calls the desired modules in turn and records information on the USELOG as follows:
  - name of user, name of file, date, clock time (in seconds) when each module is entered, and the elapsed CPU time (in seconds) since the start of the current run.

*1.6.2 Global Instructions.* Finally, the main program Omnidata sets and resets global switches and global parameters required for the selection and control of various modes of operation enumerated below. Global here means that the switches and parameters are available to all of the modules.

In the normal (or default) operating mode, the system:

- . prints lines up to a maximum of 68 characters in width;
- . reads or otherwise operates on files in the forward direction;
- . prints, after each module has finished its work, the CPU seconds in the module and the elapsed time since the initiation of the run; and
- . converses with the user in the TERSE mode (omits a listing of modules in the system and of the data items in the file).

```

* * * * *
* >@xbasic
* XBASIC R5.1      16:05:07      8 MAR 77
* >old:omnidata*
* READY
* >run
* OMNIDATA      16:05:24      8 MAR 77
*
*          *PLEASE ENTER ACCOUNT NUMBER --->? >bjbm (A)
*          *TYPE PASSWORD --->? >xyz
*          *WHICH DATA BASE DO YOU WANT --->? >fpr75
*          GOOD AFTERNOON, WELCOME TO OMNIDATA.
*          * * * NOTE — OMNIDATA KEEPS A RECORD OF WHO USED WHICH
*          *          MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*          *          FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
*          *          *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*          --->? >modules (B)
*          THE FOLLOWING MODULES ARE AVAILABLE:
*          ABRIDGE      AGGREGATE      ANALYSIS      ANNEX
*          ARRAY        ATTACH        BLANKS        BROWSE
*          CHECKSUM     COMPUTE      CONCAT        CROSSTAB
*          DESCRIBE     DICTIONARY  DISPLAY      DISTRIBUTE
*          ENCODE       EXTRACT     FETCH        FIT
*          GRAPH        KWOC        MOVE         PLAN
*          PLOT         RANDOM      REGRESS     RENAME
*          REPORT       SAVE        SCREEN      SEARCH
*          SEGMENT     SEQUENCE   SORT        STACK
*          STATIS      STATPLOTS  SUMMARY     SURVEY
*          TALLY       TRIM       UPDATE     UPDATESEQ
*          USERS
*
*          *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*          --->?      >stop
*          PROGRAM STOPPED.
*
*          TIME :      3.030
*          * * * * *

```

Figure 1.6b. This run was initiated primarily to show how the program prints out the list of modules in response to the request at [B]. If the input at [A] had been BJB,VERBOSE the module names and labels of the 111 data items would have been printed out automatically.

The normal modes of operation can be changed or reinstated by responding with one or more appropriate instructions when requested to TYPE A MODULE NAME AND/OR INSTRUCTIONS.

The following is a list of global instructions which modify the normal operating modes until restored by subsequent instructions, or otherwise altered.

**BACKWARD**

Instructs the system to read or otherwise operate on the data file from the end of the file instead of the beginning.

**COST**

Instructs the main OMNIDATA program to print out an approximate cost for work performed by each module used after the COST has been requested. This instruction can be counteracted by typing NO COST in the response to:

\*TYPE A MODULE NAME AND/OR INSTRUCTION --->? >

Since the cost is system dependent, each installation has the option of inserting the appropriate parameters to compute the cost. If that option is not exercised, the program responds with

COST = \$ SYSTEM DEPENDENT

**DATE,YYMMDD**

Provides a means of dating the output when desired.

**FORWARD**

Restores the system to operation from the front of the file.

**INTERRUPT,n**

Forces the operation to stop each time it has performed its operation on n records; prints out an appropriate message; and gives the user the choice to CONTINUE the operation, or to RESTART in the module, or to EXIT back to Omnidata in order to call a different module or enter one of the global instructions.

**LABELS**

Prints, in alphabetic order, the current names (labels) associated with the data items in the file in use.

**LENGTH**

Prints out the number of logical records in the current file.

**LIMIT,a,b**

Limits the operation of the system to a portion of the file, from record a to record b.

**MODULES**

Prints, in alphabetic order, the names of all of the Omnidata modules.

**MONITOR,n**

Prints out an appropriate message each time it has carried out its operations on n records of the file, but does not stop each time as does the INTERRUPT instruction.

- NO COST** suppresses the printing of the cost of the operation of a module.
- NO DATE** Suppresses the printing of the date if it has previously been invoked by the DATE instruction.
- NO TIME** Suppresses the printout of the CPU seconds elapsed in the run, etc., if it had been invoked by the TIME instruction.
- STORE** Allows for the automatic storage of results from certain of the modules (REPORT, SURVEY, KWOC, etc.). Each module which has a provision to store results on a file automatically generates a name for the stored file and informs the user accordingly. The STORE instruction does not carry over from module to module. It must be set each time a module is called (i.e., SURVEY, STORE, WIDTH, 100).
- TIME** Restores the system to print the elapsed CPU seconds in the run, etc., if that feature had previously been disabled by the NO TIME instruction.
- WIDTH,n** Instructs the system to allow for line widths of n characters. In demand mode, if the line length is to be set longer than 72 characters it is necessary to alert the Executive system as well as Omnidata. This is achieved by typing the Executive command @@TTY W,n any time after the run sequence. The above EXEC instruction is not necessary when running in the batch mode. The system allows for a maximum width of 132 characters.

\* \* \* N O T E S \* \* \*

```

* * * * *
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
*
* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS (A)
* --->? > width,85,labels
*
* * * * *
* THE FILE FPR75 CONTAINS DATA LABELED AS FOLLOWS:
* 98 AA 53 ac 33 ADTIT 111 AGE 68 ALC 102 APAD
* 55 APPRO 40 ATD 57 AUDAT 51 AUTH 110 C 4 CWT
* 62 CL 91 CLWOP 89 CNHS 36 COLDEG 24 CSDF 93 CSL
* 71 DAA 27 DDAY 104 DDED 35 DEG 22 DEPT 1 DIV
* 80 DLABR 48 DLIM 6 DOB 28 DOGR 30 DOPP 76 DPSQ
* 61 DRPB 50 EDA 31 EOD 32 eody 94 FAB 16 FUD
* 78 GDF 70 GLOC 18 GRADE 75 HBPN 44 HC 82 HWD
* 107 HWWAE 43 INS 23 LC 26 LEGR 88 LWOP 101 LWOPSA 67 MC
* 105 MR 5 NAME 100 NHSA 46 NOAC 99 NP 13 NTEDA 63 NTEDP
* 15 OSC 20 PAY 60 PD 74 PF 54 PFC 29 PINFO 21 PLANT
* 52 POS# 14 PP 34 PROFS 56 REM 45 RET 85 RHWSW 72 RS
* 109 SALBAS 79 SC 8 SCD 69 SCD 108 SCDY 41 SEQ
* 3 SEX 106 SK 47 SLIM 58 SON 77 SP 38 SPP
* 2 SS# 73 SSNC 9 STAT 19 STEP 17 TITLE 10 TOA
* 11 TOD 12 TOLA 65 TP 42 VP 86 WAEDWP 84 WAEDWS 87 WAEDWT
* 83 WAEPL 95 WDD 96 WDUD 97 WED 7 YOB 37 YR
*
* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* *PROGRAM STOPPED.
* * * * *

```

Figure 1.6c. A printout of the labels (in alphabetic order) associated with the file FPR75, produced in response to the LABELS request at [A]. A sign-on in the VERBOSE mode would produce automatically a list of labels in numeric order. The format of this output varies with the current Omnidata width setting and the length of the longest label.

*1.6.3 System and File Security.* It is a feature of virtually all viable computer systems that provision is made for system and file security by means of *passwords* for account numbers, *lock words* for programs, and *tape labels* for access to magnetic tapes. Once access is gained to the computer via a proper sign-on procedure, and to the Omnidata programs, the user must identify himself to the main program, Omnidata, by typing his name and a *password*. Both of these are checked against a list of authorized users and passwords resident in the particular file requested. Unless there is a specific match, the user is not allowed to operate on that file. This arrangement puts the responsibility for protecting the file upon those with responsibility to maintain the file.

In figure 1.6d we show how Omnidata refuses to allow a file to be used until the data file administrator has built into it at least one authorized user and an associated password. Figure 1.6e shows how the program responds to an improper account number, or password, or both.

The Omnidata system allows for an additional level of security since it is often necessary to allow access to many people for most, but not all, of the data items in the file. In particular, such items as grade, pay, volume of business, or margin of profit may be considered more restricted than other data items.

The Omnidata system handles this problem in an interesting way. At the time a user is accredited to a particular file by the file manager, a flag is set to indicate whether that user is a *restricted* or *unrestricted* user. Corresponding flags are attached to the sensitive data items in the file which should be denied to the restricted users.

When Omnidata recognizes a fully accredited user, it types out the labels for all of the vectors in the file and gives the user access to all of them. When a restricted user is encountered, the names of the restricted vectors are not mentioned in the printout, nor are they stored in memory, hence the user supposedly does not know that the restricted items are in the file. If he should know from other sources the names (labels) of the restricted items and asks for them, the system would not be able to comply, because it would not find those names in the label table stored in memory.

The security provided by the Omnidata system is in addition to the normal file security provided by the UNIVAC® The Executive operating system. The latter makes use of read/write keys. These keys are normally appended to the file name as follows: FNDEMO/RRR/WWW, where RRR represents the password for reading a file and WWW stands for the password to write on a file. Since Omnidata never (see ANNEX for the single exception) writes directly on the files it can access, the file request in response to the question:

\*WHICH DATA BASE DO YOU WANT --->? >

will contain only one slash (/) when the file has been protected with a read key. Once Omnidata has obtained access to the file, it deletes the read key and prints out only the name of the file. For this reason, read keys do not appear in any of the illustrations in this manual.

```

* * * * *
* >old:omnidata*
*  READY
* >run
* OMNIDATA      16:30:43      8 MAR 77
*
* *PLEASE ENTER ACCOUNT NUMBER --->? >abc
* *TYPE PASSWORD --->? >xyz
* *WHICH DATA BASE DO YOU WANT --->? >fndemo
*
* !NO ACCREDITED USERS IN YOUR FILE — IF THIS FILE
* BELONGS TO YOU, CALL PROGRAM USERS, OTHERWISE CALL
* THE DATA FILE ADMINISTRATOR.
* TIME:      1.985
* * * * *

```

Figure 1.6d. Here we see how Omnidata refuses to allow a file to be used until the data file administrator has built into it at least one authorized user and an associated password.

```

* * * * *
* *PLEASE ENTER ACCOUNT NUMBER --->? >abc
* *TYPE PASSWORD --->? >123
* *WHICH DATA BASE DO YOU WANT --->? >fpr75
* WRONG PASSWORD - RETYPE --->? >xyz (A)
* GOOD AFTERNOON, WELCOME TO OMNIDATA.
* * * NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* PROGRAM STOPPED.
* * * * *
* *PLEASE ENTER ACCOUNT NUMBER --->? >bbc
* *TYPE PASSWORD --->? >xyz
* *WHICH DATA BASE DO YOU WANT --->? >fpr75
* WRONG ACCOUNT NUMBER - RETYPE --->? >abc (B)
* GOOD AFTERNOON, WELCOME TO OMNIDATA.
* * * NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* PROGRAM STOPPED.
* * * * *
* *PLEASE ENTER ACCOUNT NUMBER --->? >aaa (A)
* *TYPE PASSWORD --->? >zzz (B)
* *WHICH DATA BASE DO YOU WANT --->? >fpr75
*
* !YOU HAVE NOT BEEN ACCREDITED TO USE FPR75
* PLEASE CALL THE DATA FILE ADMINISTRATOR
*
* TIME : 2.123
* * * * *

```

Figure 1.6e. In the upper portions it is shown how the main Omnidata program allows the user to correct improper inputs of passwords [A] or account numbers [B]. If both are in error, the Omnidata run is terminated as shown in the bottom third of this figure.

*1.6.4 Error Recovery.* Any good interactive computer system, depending as it does on dialogue, must have ample provisions for coping with typing errors or other improper responses to input requests. Such errors can result from poor typing, unfamiliarity with the keyboard, or a misunderstanding of the request. Even experienced typists and otherwise knowledgeable users sometimes give the right answer to the wrong question. It is therefore important to explain to what extent the Omnidata system tolerates errors at various stages in the solution of a problem.

We consider first the case where an error is discovered in a line before the carriage return key is depressed. We have two options here:

- a) we can backspace and type over the errors. The location of the backspace key varies from terminal to terminal.

On the UNIVAC® 1108 the CNTL Z is equivalent to the backspace.

- b) we can cancel the entire line of input by typing X with the control key depressed and then retyping the entire line.

If the error is not discovered until after the carriage return key is depressed, we must rely on how Omnidata treats the error. The consequences of a typing error depend upon its nature, in which module it occurs, and where in the dialogue it occurs. For example, in the main module Omnidata

- a) a misspelled or improper file name results in the message:  
FILE CAN NOT BE ASSIGNED  
FILENAME NOT IN MASTER FILE DIRECTORY; and
- b) an improper account number or password results in the message:  
? WRONG ACCOUNT NUMBER - RETYPE--->?

or

- ? WRONG PASSWORD - RETYPE--->?

If both the account number and the password are incorrect, the run is terminated after printing the message

! YOU HAVE NOT BEEN ACCREDITED TO USE *FILENAME*  
PLEASE CALL THE DATA FILE ADMINISTRATOR

If the typo is in one of the global commands discussed in section 1.6.2, such as DATE, LENGTH, TIME, BACKWARD, etc., it is ignored. The desired action is not taken and the rest of the response is carried out, if proper, after the following message:

XYZ IS NOT A LEGITIMATE OPTION!

If such a typing error would invalidate the results, the user can resort to the interrupt option discussed in section 3.3.

The consequences of input errors in other modules vary from module to module and are illustrated in appropriate sections to follow.

## 1.7 References

1. Davis, P.J., and Rabinowitz, P., Advances in orthonormalizing computation, in *Advances in Computers*, Vol. 2, pp. 55-133, Academic Press, New York, NY (1961).
2. Walsh, P., *Ortho*, *Communications ACM*, Vol. 5, pp. 511-513 (1962).
3. Hilsenrath, J., and Galler, G.M., OMNIFORM I: A general-purpose machine program for the calculation of functions given explicitly in terms of one variable, *National Bureau of Standards Technical Note 125* (May 1962).
4. Cameron, J.M., and Hilsenrath, J., Use of general-purpose coding systems for statistical analysis, in *Proceedings of the I.B.M. Scientific Computing Symposium* (October 21-23, 1963), pp. 281-299, published by I.B.M., White Plains, NY (1965).
5. Hilsenrath, J., Ziegler, G.G., Messina, C.G., Walsh, P., and Herbold, R.J., OMNITAB: A computer program for statistical and numerical analysis, *National Bureau of Standards Handbook 101* (March 4, 1966; reissued January 1968 with corrections).
6. Beam, A.E., and Hilsenrath, J., PRECISE: A multiple precision version of OMNITAB, *National Bureau of Standards Technical Note 556* (June 1968).
7. Hogben, D., Peavy, S.T., and Varner, R.N., OMNITAB II: User's Reference Manual, *National Bureau of Standards Technical Note 552* (October 1971).
8. Longley, J.W., An appraisal of least-squares programs for the electronic computer from the point of view of the user, *Journal of the American Statistical Association*, Vol. 62, pp. 819-841 (1967).
9. Wampler, R.H., An evaluation of linear least-squares computer programs, *Journal of Research of the National Bureau of Standards*, Vol. 73B, pp. 59-90 (1969).
10. Wampler, R.H., A report on the accuracy of some widely used least-squares computer programs, *Journal of the American Statistical Association*, Vol. 65, pp. 549-565 (1970).
11. Wampler, R.H., Some recent developments in linear least-squares computations, in *Proceedings of the Computer Science and Statistics Sixth Annual Symposium on the Interface*, (October 16-17, 1972), pp. 94-109, University of California at Berkeley.
12. Hogben, D., and Peavy, S.T., OMNITAB II: User's Reference Manual 1977 Supplement, NBSIR 77-1276.

13. Jowett, D., Chamberlain, R.L., and Mexas, A.G., OMNITAB: A simple language for statistical computations, *J. Statis. Comput. Simul.*, Vol. 1, pp. 129-147 (1972).
14. Donnay, J.D.H., and Ondik, H.M. Eds., *Crystal Data Determinative Tables*, Third Edition in two volumes, The Joint Committee on Powder Diffraction Standards, Philadelphia, PA (1972).
15. Mighell, A.D., Ondik, H.M., and Molino, B.B., *Crystal Data Space-Group Tables*, *Journal of Physical and Chemical Reference Data*, Vol. 6, No. 3 (1977).
16. Cardenas, A.F., Evaluation and selection of file organization—A model and system, *Communications of the ACM*, Vol. 16, pp. 540-548 (1973).
17. Ku, H. H., A Users' Guide to the OMNITAB Command "STATISTICAL ANALYSIS", National Bureau of Standards Technical Note 756 (March 1973).

\* \* \* N O T E S \* \* \*

## 2. Conversion of Conventional Files into Omnidata Format

The Omnidata system can handle fixed-field data files that have been placed in sectors on mass storage (magnetic discs or drums) in such a way as to allow the user to access a particular data item not by its location on the disc, but by asking for it by name. In order to do this, it is necessary for the file to carry information (pointers) in header sectors which direct the system to the location of the desired data items on the mass storage device.

The conversion of an existing fixed-field data file to Omnidata format is carried out by an independent XBASIC program called DEFINE. This program gets its information about the record layout from a small XBASIC file in which the location of the data items in the records is shown together with the labels (names) by which they will be identified in subsequent use of that data file. Some typical label files are shown in figure 2a. If it is more convenient for the user, this information can be entered on-line during the running of the DEFINE program rather than having it previously stored on an XBASIC file.

### 2.1 The DEFINE Program

This is a separate XBASIC program which takes a conventional unblocked fixed-field file of Fieldata characters and converts it to Omnidata format using instructions given in an XBASIC file as shown in figure 2.1a.

Thus, the first operation to be performed in the defining process is to establish the fields, or data items, one will want the Omnidata system to be able to identify. For each data element, six pieces of information are needed as is shown in figure 2.1a. These are:

- 1) the name of the data item (as short and mnemonic as possible);
- 2) the physical record in the original file in which the data item appears (e.g.,—if original data is on cards, is this element on card 1, card 2, or card 3,...?);
- 3) the character position in the original record in which the data item begins;
- 4) the character position in the original record in which the data item ends;
- 5) '1' if the item is numeric, '0' if it is alphabetic or alphanumeric;
- 6) '1' if restrictions are to be placed on the use of this data item, '0' if everyone accessing the data base as a whole can also access this data item.

```

* * * * *
* NLABELDATE 10:11:03 8 SEP 76
*
* 10 * INST,1,278,278,1,0
* 20 *SS ,1,1,9,0,0
* 30 * SEX,1,11,11,0,0
* 40 * CIT,1,12,12,0,0
* 50 * NAME,1,13,48,0,0
* 60 * DOB,1,49,54,0,0
* 65 * MOB,1,49,50,0,0
* 70 * YCB,1,53,54,1,0
* 80 * SCD,1,60,65,0,0
* 90 * STAT,1,72,72,0,0
* 100 * TOA,1,73,73,0,0
* 110 * TOD,1,74,74,0,0
* 120 * TOLA,1,75,78,0,0
* 130 * NTE,1,79,84,0,0
* 140 * PP,1,120,121,0,0
* 150 * CSC,1,122,126,0,0
* 160 * PC,1,127,128,1,0
* 170 * TITLE,1,129,260,0,0
* 180 * GRADE,1,1261,262,1,1
* 190 * STEP,1,263,264,1,1
* 200 * PAY,1,265,269,1,1
* 210 * DIV,1,278,280,1,0
* 220 * SEC,1,278,282,1,0
* 230 * LC,1,283,291,0,0
*
* 1040 * DDED,1,658,663,0,0
* 1050 * MR,1,664,664,0,0
* 1060 * SK,1,565,635,0,0
* 1070 * HMMAE,1,867,867,0,0
* 1080 * SCDY,1,64,65,1,0
* 1090 * SALBAS,1,270,270,0,0
* 1100 * C,1,493,494,0,0
* 1102 * EDLEV,1,687,388,1,0
*
* * * * *

```

```

LIST 10:13:29
NLABELS
10 * NUM,1,1,13,0,0
20 * EXT,1,14,17,1,0
30 * NAME,1,16,47,0,0
40 * DIV,1,48,50,1,0
50 * SEC,1,48,53,1,0
60 * BLDG,1,54,58,0,0
70 * ROOM,1,59,53,0,0
80 * MBLDG,1,64,68,0,0
90 * WROCM,1,69,73,0,0
100 * STREET,1,74,101,0,0
110 * CITY-ST,1,102,121,0,0
120 * ZIP,1,123,127,1,0
130 * EXCH,1,131,133,1,0
140 * PHONE,1,131,138,0,0
    * UNADD,1,122,122,0,0
    * UNADD,1,139,139,0,0

```

```

LABJPCRD 15:20:12
10 * NUM1,1,3,1,0
20 * JOURNAL,1,4,13,0,0
30 * YR,1,14,17,1,0
40 * VOL,1,18,25,0,0
50 * NO,1,26,32,0,0
60 * PG1,33,39,0,0
70 * ID,1,18,39,0,0
80 * AUTHORS,1,40,239,0,0
90 * TITLE,1,240,439,0,0
100 * KEYWORDS,1,440,639,0,0
110 * PROPERTIES,1,640,639,0,0
900 * END,0,0,0,0,0
999 END

```

Figure 2a. Some examples of XBASIC files used to instruct the DEFINE program to convert conventional data files to Omnidata format. The numbers to the left of the asterisk (\*) are arbitrary XBASIC line numbers. The numbers following the mnemonics (labels) assigned to the data items tell in which record the data item is to be found. The next two numbers locate the data item in the record. The fourth number indicates if the data item is alphabetic or numeric, and the last number is a security flag.

The above information should be entered as a program under the XBASIC compiler. Each data item should be a separate line consisting of a line number followed by an '\*' (indicating data rather than an instruction), after which the six pieces of information are entered separated by commas. After each element has been entered on a separate line, the following 2 lines

```
998*END,0,0,0,0,0
999END
```

are needed to conclude the procedure. Finally, by giving the command FSAVE:LABELS the above information is stored in a temporary file called 'LABELS', which can be accessed during the defining process.

To illustrate the above, let us consider a file of names and phone numbers. Each record is on a punched card which contains the name in the first 20 columns (positions) and the extension, which is in positions 21-24. The following procedures would serve as the field definitions:

```
@RUN
@XBASIC
10*NAME,1,1,20,0,0
20*EXT,1,21,24,1,0
998*END,0,0,0,0,0
999END
FSAVE:LABELS
```

Several additional points need to be made at this time. To begin, the above line numbers (10,20,...) are arbitrary and could just as well have been 1,2, or 100,200, or 2,4, etc. The labels will be listed in Omnidata in the order of ascending line numbers, but they do not necessarily have to be input in the order the fields occur in the file. EXT could have been entered before NAME. Another point to be made is that a field can be multiply defined. For example, at the same time the entire name field is defined as characters 1-20, there could be a subfield LASTNAME of, say, characters 1-10 and a subfield FIRSTNAME consisting of positions 11-20. Figure 2.1a shows the information required to define the elements of an annotated bibliographic file and the subsequent information required by the DEFINE module.

Storing the record layout information on a file as described and illustrated above has the advantage that the user need only type it once, and it will be available for all subsequent DEFINE runs when needed. This is useful if, say, the list of labels is lengthy and one will be defining identical files periodically. However, if the list of labels is short and if the user would prefer, the DEFINE program allows the user to enter the

record layout information on-line while the program is running. During the DEFINE run, for both cases (when the data is on cards or in a file) as described below, the user will be asked to:

\*TYPE F(ILE) FOR STORED LABELS, OR I(NPUT) TO ENTER--->? >

If the record layout is stored as above, the user should answer F or FILE. An answer of I or INPUT will prompt the program to start asking for information for

\*LABEL1--->? >

\*LABEL2--->? >

\*LABEL3--->? >

etc.

The data required is the same as above, excluding the XBASIC line number and the asterisk (\*). Thus, 'NAME,1,1,20,0,0' would be an appropriate answer to the above request for LABEL1, LABEL2, etc. The response 'END,0,0,0,0,0' is required to end the operation.

Upon encountering the END,0,0,0,0,0 response the DEFINE program prints out the labels as it had received them, with sequential line numbers, and asks

\*DO YOU WISH TO CHANGE ANYTHING--->? >

If the user answers 'NO', the DEFINE program continues. A response of 'YES' evokes the request:

\*TYPE LINE NUMBER, NEW OR CORRECTED LINE--->? >

One can delete lines, add lines, or correct lines in the following manner:

- a) DELETE 1,5,12 will delete lines 1, 5, and 12.
- b) 12,'YOB,1,5,6,1,0' will replace whatever was line 12 with the sequence 'YOB,1,5,6,1,0'.
- c) 13.5,'MOB,1,3,4,1,0' will insert the line 'MOB,1,3,4,1,0' between lines 13 and 14.

When all corrections have been entered (signalled by 'END' as a response to the above question), the user is asked if he would like a new listing of the labels. When he answers 'NO' the DEFINE program continues.

Once the field specification is accomplished, as explained above, the actual defining program, DEFINE, can be run. This program is designed to be interactive and prompts the user as shown in figure 2.1a to input the various parameters which are required. The program is invoked, while still under the XBASIC system, by typing

OLD:DEFINE\*

then, when the computer responds with a 'READY', the user types 'RUN'.

The first question which this program asks the user is:

\*IS YOUR DATA ON C(ARDS) OR F(ILE) --->? >

This determines which of the two modes of operation will be followed, and they are different enough to warrant separate discussions. Please note that a response of 'CARDS', or 'C', does not necessarily have to refer to the physical medium of punched cards. It could just as easily be card images stored on mass storage.

## **2.2 File Definitions When the Original Data Is in File Format on Mass Storage**

If the answer to the above question was 'FILE', or 'F', the defining program assumes that the data are presently in UNIVAC file format and asks for the name of this file through the question

\*TYPE ORIGINAL FILE NAME --->? >

The program attempts to assign the file named, and if the assign fails, the define operation stops.

The next two questions which are asked are interrelated:

\*TYPE NUMBER OF PHYSICAL RECORDS PER LOGICAL RECORD--->? >

\*TYPE LENGTH OF EACH PHYSICAL RECORD--->? >

Say, for example, each record occupies one sector of the original file. Then the answer to the first question above would be '1', and the answer to the second would be '168'. If, on the other hand, each record occupies two sectors, your answers could be '1' and '336', or '2' and '168', depending on how you defined your fields above. In an unblocked file, the length of each physical record should be a multiple of 168 (the length of one sector).

Next the user is prompted to

\*TYPE NAME FOR NEW FILE--->? >

```

* * * * *
*
* >old:labjpcrd
*   READY
* >list
*   LABJPCRD      09:35:37      21 MAR 77
*
*   10 * NUM,1,1,3,1,0
*   20 * JOURNAL,1,4,13,0,0
*   30 * YR,1,14,17,1,0
*   40 * VOL,1,18,25,0,0
*   50 * NO,1,26,32,0,0
*   60 * PG,1,33,39,0,0
*   70 * ID,1,18,39,0,0
*   80 * AUTHORS,1,40,239,0,0
*   90 * TITLE,1,240,439,0,0
*  100 * KEYWORDS,1,440,639,0,0
*  110 * PROPERTIES,1,640,839,0,0
*  120 * FLAG,1,840,840,0,0
*  900 * END,0,0,0,0,0
*  999 END
*
* >fsave:labels
*   READY
* >old:define
*   READY
* >run
*
*   DEFINE      09:36:31      21 MAR 77
* * * * *

```

Figure 2.1a. Here we see an XBASIC file called LABJPCRD containing the labels and record layout for an annotated bibliographic file called JOURNALS. In order to use that file in the DEFINE module, it is necessary to SAVE it temporarily under the name LABELS as at [A]. Now we call and RUN the DEFINE module at [B]. The dialogue that follows is explained in the text. The response at [C] notifies this module to get its information from the saved file shown above. If the response at [D] is YES, the module is prepared to carry on an extensive arithmetic operation which we choose to by-pass here. The request at [E] is not made by the DEFINE module; it is made by the module called USERS to which the program has switched (chained) automatically to allow the data file administrator to accredit one or more users to this newly defined file. See subsequent figures and section 6.8 for a discussion of the USERS module.

```

* * * * *
* >fsave:labels
*   READY
* >old:define
*   READY
* >run
*
* DEFINE          09:36:31    21 MAR 77
*
*   *IS YOUR DATA ON C(ARDS) OR F(ILE) --->? >file
*   *TYPE ORIGINAL FILE NAME
* ? >journals
*   *TYPE NUMBER OF PHYSICAL RECORD
* ? >1
*   *TYPE LENGTH OF EACH PHYSICAL RECORD
* ? >840
*   *TYPE NAME FOR NEW FILE
* ? >ftjournals
*   *HOW MANY TRACKS
* ? >128
*   *TYPE F(ILE) FOR STORED LABELS, OR I(NPUT) TO ENTER --->? >f
* LABELS DONE    1.0038
*   *DO YOU WANT CHECKSUMS?
* ? >no
*   *TYPE # RECORDS
* ? >48
*   *TYPE NUMBER OF HEADER SECTORS, OR 0 FOR NONE --->? >0
*   *TYPE NUMBER OF HEADER WORDS IN RECORD, OR 0 FOR NONE --->? >0
*
*   *WHICH DATA BASE DO YOU WANT --->? >ftjournals
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >add
*   *TYPE NAME OR ACCOUNT NUMBER --->? >abcd
*   *TYPE PASSWORD --->? >xyz
*   *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
*   *TYPE NAME OR ACCOUNT NUMBER --->? >end
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >stop
* PROGRAM STOPPED.
* * * * *

```

The diagram consists of three circled letters: C, D, and E. A line from C points to the prompt '>file'. A line from D points to the response '>no'. A line from E points to the prompt '>ftjournals'.

Figure 2.1a (concluded). An illustration of file definition when both the original data base and the record layout are in file format on mass storage as indicated at [C].

This filename should consist of from 1 to 12 characters and may be taken from the set A-Z, 0-9, -, and \$, which is consistent with UNIVAC requirements. If one is intentionally writing over an already existing file, the file name should be followed by 'A', signifying that it is an Already assigned file. If there is no 'A', the program assumes it is to assign a new file and asks the user

\*HOW MANY TRACKS --->? >

For an appropriate answer to this question, it is usually sufficient merely to add 10 or 20 tracks to the number of tracks which the original file occupied. These additional tracks allow the DEFINE program to store the labels and other header information necessary for normal operation of the Omnidata system. If, however, the number of tracks in the original file is not known, the answer to the above question can be computed by multiplying the number of words in a record (in multiples of 28) by the number of records, dividing this by 64, and finally adding 10 or 20 tracks, as above.

Assuming the file cataloguing operation was successful, the program now reads through the label information stored previously through the FSAVE:LABELS command or asks for the labels as input. It then stores this in the header section of the newly catalogued file.

The next question the user is asked is

\*DO YOU WANT CHECKSUMS--->? >

This is one method of providing internal file security, but we must caution the user that to compute these checksums is a rather costly operation. In essence, what is done is that the numeric equivalent of each character in the record is obtained. Those in the odd positions are summed, and then those in the even positions are summed. These two sums are stored in each record. At a later time these checksums may be recomputed, and should the results not be identical to the originally computed sums, the file manager can be alerted that the data has been altered. If, during the defining process, the user asks for these checksums, one further question is asked:

\*TYPE: SECTOR, BEGINNING, ENDING OF CHECKSUMS--->? >

The user must designate blank positions in the original record where these sums can be stored. For example, perhaps the original record has purposely been defined as 2 strings of 168 characters each, but the second string really has information only through position 100. Then the answer to the above question could be to store these checksums in sector 2 in positions 101-168.

The answer to the question

\*TYPE # RECORDS--->? >

is simply the number of records in the original file which the user desires to define into Omnidata format.

Two more inputs are requested from the user before the actual definition is done. These questions relating to the structure of the original file are:

\*TYPE NUMBER OF HEADER SECTORS, OR 0 FOR NONE--->? >

and

\*TYPE NUMBER OF HEADER WORDS IN RECORD, OR 0 FOR NONE--->? >

Ordinarily the answers to both of these questions will be '0', and the DEFINE program will start reading the data at the very beginning of the file. It will assume that valid information begins in the first word of each record. An example which would prove to be an exception to this is, say, a file created by a COBOL program, in which there are both header sectors at the beginning of the entire file, and header words at the beginning of each record. The DEFINE program must be told how much information to skip in order to get at the actual data in the file.

Following this, the actual defining is accomplished. The records are read from the original file, and after certain operations are performed on them (such as stripping out header words or calculating checksums if desired), they are written out to the new file in Omnidata format. During this operation, a statement is printed every time 100 records are read to give an indication of the progress of the procedure.

When all records have been satisfactorily read, the DEFINE program chains to the program USERS to allow the file manager to input account numbers and passwords for those individuals he wishes to validate as legitimate users of that particular data file. Since the program USERS is also a valid Omnidata module, the detailed discussion of its operation is handled under the section in this manual describing the various modules. At the conclusion of this run, the user should have a file in Omnidata format on which he can perform all of the available Omnidata operations.

Figure 2.2a shows how this module permits corrections when the record definition is supplied from the keyboard.

```

* * * * *
* >old:define
*   READY
* >run
*
*   DEFINE          09:39:48    21 MAR 77
*
*   *IS YOUR DATA ON C(ARDS) OR F(ILE) --->? >file
*   *TYPE ORIGINAL FILE NAME
* ? >journals
*   *TYPE NUMBER OF PHYSICAL RECORD
* ? >1
*   *TYPE LENGTH OF EACH PHYSICAL RECORD
* ? >840
*   *TYPE NAME FOR NEW FILE
* ? >ftjournals
*   *HOW MANY TRACKS
* ? >128
*   *TYPE F(ILE) FOR STORED LABELS, OR I(NPUT) TO ENTER --->? >I
*   *LABEL 1 --->? >num,1,1,3,1,0
*   *LABEL 2 --->? >journal,1,4,13,0,0
*   *LABEL 3 --->? >yr,1,14,17,1,0
*   *LABEL 4 --->? >vol,1,19,25,0,0
*   *LABEL 5 --->? >no,1,26,32,0,0
*   *LABEL 6 --->? >pg,1,33,39,0,0
*   *LABEL 7 --->? >authors,1,40,239,0,0
*   *LABEL 8 --->? >title,1,240,439,0,0
*   *LABEL 9 --->? >keywords,1,440,639,0,0
*   *LABEL 10 --->? >properties,1,640,839,0,0
*   *LABEL 11 --->? >flag,1,840,840,0,0
*   *LABEL 12 --->? >end,0,0,0,0
* FOLLOWING ARE THE LABEL LINES YOU HAVE ENTERED:
*   1          NUM,1,1,3,1,0
*   2          JOURNAL,1,4,13,0,0
* * * * *

```

Figure 2.2a. Here we have a record of a file definition operation when the original data base is in file format on mass storage as indicated at [A] and the information for the record layout is supplied from the keyboard as indicated at [B]. Because on-line data entry entails the risk of error, the module provides a clean listing at [C] and an opportunity at [D] to make corrections, additions, or deletions.

```

*****
*
* 3          YR,1,14,17,1,0
* 4          VOL,1,19,25,0,0
* 5          NO,1,26,32,0,0
* 6          PG,1,33,39,0,0
* 7          AUTHORS,1,40,239,0,0
* 8          TITLE,1,240,439,0,0
* 9          KEYWORDS,1,440,639,0,0
* 10         PROPERTIES,1,640,839,0,0
* 11        FLAG,1,840,840,0
*
* *DO YOU WISH TO CHANGE ANYTHING --->? >yes
* *TYPE LINE NUMBER, NEW OR CORRECTED LINE --->
*
* ? >4,'vol,1,18,25,0,0'
* ? >6.5,'id,1,18,39,0,0'
* ? >end
*
* (F) *DO YOU WANT A CLEAN LISTING OF YOUR LABELS --->? >yes
* FOLLOWING ARE THE LABEL LINES YOU HAVE ENTERED:
* 1          NUM,1,1,3,1,0
* 2          JOURNAL,1,4,13,0,0
* 3          YR,1,14,17,1,0
* 4          VOL,1,18,25,0,0
* 5          NO,1,26,32,0,0
* 6          PG,1,33,39,0,0
* 7          ID,1,18,39,0,0
* 8          AUTHORS,1,40,239,0,0
* 9          TITLE,1,240,439,0,0
* 10         KEYWORDS,1,440,639,0,0
* 11        PROPERTIES,1,640,839,0,0
* 12        FLAG,1,840,840,0,0
*
* *DO YOU WISH TO CHANGE ANYTHING --->? >no
* LABELS DONE          1.0512
* *DO YOU WANT CHECKSUMS?
* ? >no
* *TYPE      RECORDS
* ? >48
*
* *TYPE NUMBER OF HEADER SECTORS, OR 0 FOR NONE --->? >0
* *TYPE NUMBER OF HEADER WORDS IN RECORD, OR 0 FOR NONE --->? >
*
*
* *WHICH DATA BASE DO YOU WANT --->? >fti journals
*
*****

```

The diagram consists of several callouts labeled C through H, each enclosed in a circle. Callout C points to lines 3 through 11. Callout D points to the prompt '\*DO YOU WISH TO CHANGE ANYTHING --->? >yes'. Callout E points to the input lines '? >4,\'vol,1,18,25,0,0\'' and '? >6.5,\'id,1,18,39,0,0\'' and the prompt '? >end'. Callout F points to the prompt '\*DO YOU WANT A CLEAN LISTING OF YOUR LABELS --->? >yes'. Callout G points to lines 4 through 7. Callout H points to the prompt '\*DO YOU WISH TO CHANGE ANYTHING --->? >no'.

Figure 2.2a (concluded). Here we see at [E] how to insert a line and that in order to correct a mistake in a line it is necessary to retype it. After the usual way of terminating an open-ended input at [F], we see from the second clean listing at [G] that line four has indeed been corrected and that a line has been inserted following line six. The response at [H] causes the module to proceed with the defining operation as shown in the previous figure.

### 2.3 File Definition When the Original Data Is on Cards or Card Images on Mass Storage

In figure 2.3a we see how this module operates when the user has responded at [A] that his original data is on cards. The DEFINE program asks for the number of cards per logical record, and then request:

\*TYPE LOCATION OF SEQUENCE NUMBER OR 0 IF NONE --->? >

The answer to this question should be pairs of numbers, each pair indicating the starting position and ending position of the record number for each card type in turn. For example, if there are 3 cards in a record, and the sequence number on cards 1 and 3 appears in columns 1-5, and those on card 2 appear in columns 76-80, the answer to the above would be:

1,5,76,80,1,5

It should be noted that if the record number appeared in the same location on all 3 cards (say, in columns 1-5), the question could be answered either by

1,5,1,5,1,5

or by

1,5

In the latter case the program assumes that the one pair of numbers refers to all of the cards.

If the locations of the record numbers are given, the cards for each record are read and are checked to make sure the numbers agree. If not, an error message is printed and that record is skipped.

Next the user is asked to

\*TYPE COLUMNS ON EACH CARD TO BE INCLUDED --->? >

Once again, pairs of numbers are required giving the starting position and ending position on each card in turn. The same convention as above holds. That is, if there are three cards per record and only one pair of numbers is given in response to the last question, that pair is considered to be applicable to all cards in the record. In response to either of the above questions, if an odd number of entries are given an error message is printed, and the user is asked to reenter the answer.

```

* * * * *
* >OLD:LABJPCRD
* > READY
* >FSAVE:LABELS
* > READY
* >OLD:DEFINE
* > READY
* >RUN
*
* DEFINE 14:14:33 21 MAR 77
* *IS YOUR DATA ON C(ARDS) OR F(ILE) --->? >CARDS [A]
* *TYPE NUMBER OF CARDS PER RECORD --->? >11
* *TYPE LOCATION OF SEQUENCE NUMBER OR 0 IF NONE --->
* ? >0
* *TYPE COLUMNS ON EACH CARD TO BE INCLUDED --->
* ? >1,80
* *TYPE F(ILE) FOR STORED LABELS, OR I(NPUT) TO ENTER ---> >F
* *TYPE NAME FOR NEW FILE
* ? >FTJOURNALS
* *HOW MANY TRACKS
* ? >128
* LABELS DONE 1,0944
* *DO YOU WANT CHECKSUMS?
* ? >NO
* *TYPE RECORDS [D]
* ? >48
* ? > @ADD JOURNALS.INDEX
*
* *WHICH DATA BASE DO YOU WANT --->? >FTJOURNALS
*
* *TYPE ADD, DELETE, CHANGE, OR LIST --->? >ADD
* *TYPE NAME OR ACCOUNT NUMBER --->? >BJBM
* *TYPE PASSWORD --->? >OK
* *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
* *TYPE NAME OR ACCOUNT NUMBER --->? >END
* *TYPE ADDE, DELETE, CHANGE, OR LIST --->? >LIST
*
* NAME OR ACCOUNT NO. PASSWORD FLAG [F]
* -----
* BJBM ok O
* * * * *

```

Figure 2.3a. Here we have a record of the dialogue required to define a card-image data base on mass storage using a catalogued X BASIC file containing the labels and record layout. Since we indicated at [A] that each logical record contained 11 card types, we are given the opportunity at [B] and [C] to supply as many as 11 number pairs. See the text for responses other than that at [D]. The dialogue at [E] with the USERS module accredits a user BJBM with password OK, and the LIST request at [F] shows that the information has indeed been entered in the original file.

```

*****
* OLD:DEFINE
* READY
* >RUN
*
* DEFINE 14:19:01 21MAR 77
*
*
* IS YOUR DATA ON C(ARDS) OR F(ILE) --->? >CARDS
* TYPE NUMBER OF CARDS PER RECORD --->? >11
* TYPE LOCATION OF SEQUENCE NUMBER OR 0 IF NONE--->?
? >0
* TYPE COLUMNS ON EACH CARD TO BE INCLUDED --->?
? >1,80
* TYPE F(ILE) FOR STORED LABELS, OR I(NPUT) TO ENTER --->? >i
* LABEL 1 --->? >num,1,1,3,1,0
* LABEL 2 --->? >journal,1,4,13,0,0
* LABEL 3 --->? >yr,1,14,17,1,0
* LABEL 4 --->? >vol,1,18,25,0,0
* LABEL 5 --->? >end,0,0,0,0,0
* FOLLOWING ARE THE LABEL LINES YOU HAVE ENTERED:
* 1 NUM,1,1,3,1,0
* 2 JOURNAL,1,4,13,0,0
* 3 YR,1,14,17,1,0
* 4 VOL,1,18,25,0,0
* DO YOU WISH TO CHANGE ANYTHING --->? >no
* TYPE NAME FOR NEW FILE
? >ftijournals
* HOW MANY TRACKS
? >128
* LABELS DONE .9814
* DO YOU WANT CHECKSUMS?
? >NO
* TYPE RECORDS
? >48
? >@ ADD JOURNALS,INDEX
*
* WHICH DATA BASE DO YOU WANT --->? >ftijournals
*
* TYPE ADD, DELETE, CHANGE, OR LIST --->? >add
* TYPE NAME OR ACCOUNT NUMBER --->? >bjbm
* TYPE PASSWORD --->? >ok
* TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
* TYPE NAME OR ACCOUNT NUMBER --->? >stop
* PROGRAM STOPPED.
*****

```

**A** points to the prompt: `>CARDS`

**B** points to the prompt: `>11`

**C** points to the prompt: `>i`

**D** points to the prompt: `>@ ADD`

Figure 2.3b. Here we see the dialogue required to define a data base in card-image format on mass storage via on-line input of the labels and their location in the record. The comments in the previous figure apply here as well.

Following this, the labels are either read in from the stored file, or the user is asked to input them as described earlier.

After this, the dialogue for the DEFINE program resumes with

\*TYPE NAME FOR NEW FILE --->? >

and continues along the same lines as described earlier when the original data was in file format. The only subsequent difference is that answers to the two questions

\*TYPE NUMBER OF HEADER SECTORS, OR 0 for NONE --->? >

and

\*TYPE NUMBER OF HEADER WORDS IN RECORD, OR 0 FOR NONE --->? >

are not needed; therefore, the questions are not asked.

When the file assignment has been successfully accomplished, and the header sectors determined from the appropriate record layout information and number of records, the DEFINE program is ready for the card input. The user is prompted, may enter card images on-line, may do an @ADD of an element in which card images have previously been stored, or may actually have a box of cards, if running in the batch mode.

After receiving enough card images to satisfy the given number of records, the program chains to USERS, as described earlier.

\* \* \* N O T E S \* \* \*

### 3. Searching, Reporting, and Updating of Omnidata Files

Four of the Omnidata modules have been selected for discussion here since they represent a minimum set to be understood and mastered in order to carry out the routine data retrieval and data file maintenance normally required of computerized data systems.

- DISPLAY** Provides a quick look at any or all of the data items in specified records in the current file. Each data item is suitably labeled. (See also BROWSE.)
- REPORT** Provides for either ad hoc or periodic report generation with considerable flexibility for titles, footnotes, columnar spacing, and multilevel column headings.
- SEARCH** Allows for sophisticated data retrieval from Omnidata files, including: Boolean logic, anchored or unanchored partial string searches (prefixes, suffixes, roots, stems, etc.), exact or relational matches, matches within numerical ranges, ignoring of specific intermediate characters, etc.
- UPDATE** Provides a facile way to correct or otherwise modify individual data items either record by record or in a global manner over all records satisfying a specified criterion.

Other modules to perform arithmetic operations; numerical, statistical and graphical data analysis; and a variety of numerical and logical data classifications are treated in sections 5 and 6. Those sections also include a discussion of more advanced features of the modules described here.

In figure 3a we illustrate the start of a series of Omnidata runs which will explain the features of the above modules. This figure illustrates how the system reacts when the input at [A] contains the word VERBOSE. In this talkative mode, the main Omnidata program automatically prints out the labels for all of the data elements in the requested file, as well as the names of all of the Omnidata modules.

```

*****
* OMNIDATA 14:08:55 19 SEP 77 *
*
* PLEASE ENTER ACCOUNT NUMBER -->? >bjbm,verbose *
* TYPE PASSWORD -->? >ok *
* WHICH DATA BASE DO YOU WANT -->? >fndemo *
*
* GOOD AFTERNOON, WELCOME TO OMNIDATA *
* * * NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH * *
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY *
*
* FILE FNDEMO CONTAINS 112 DATA ITEMS FOR 500 RECORDS. *
*
* THE FILE FNDEMO CONTAINS DATA LABELLED AS FOLLOWS: *
* 1 DIV 2 SS 3 SFX 4 CIT *
* 5 NAME 6 DOB 7 YOB 8 SCD *
* 9 STAT 10 TCA 11 TOD 12 TOLA *
* 13 NTEDA 14 PP 15 OSC 16 FC *
* 17 TITLE 18 GRADE 19 STEP *
* 21 PLANT 22 DEPT 92 CAWOL *
* 25 CSDTO 95 WDD 96 WDUD *
* 29 PINFO 98 AA 99 NP 100 NHSA *
* 102 APAD 103 FUD 104 DDED *
* 105 MR 106 SK 107 HWWAE 178 SCDY *
* 109 SALBAS 110 C 111 AGE 112 CAGE *
*
* THE FOLLOWING MODULES ARE AVAILABLE: *
* SEARCH REPORT SORT DISTRIBUTE *
* COMPUTE SUMMARY CROSSTAB ENCODE *
* FETCH SAVE RANDOM RENAME *
* ATTACH BROWSE EXTRACT UPDATE *
* ANALYSIS MOVE ABRIDGE TALLY *
* STATIS PLAN CONCAT SEQUENCE *
* AGGREGATE DESCRIBE ARRAY DISPLAY *
* SEGMENT PLOT GRAPH REGRESS *
* SCREEN FIT CHECKSUM UPDATESQ *
* STACK TRIM KWOC SURVEY *
* USERS DICTIONARY ANNEX STATPLOTS *
* BLANKS *
*****

```



Figure 3a. Here we see that the labels for all of the data elements in the file FNDEMO, as well as the names of the available modules, are printed out automatically because we entered in the VERBOSE mode at [A]. The labels are printed in the numerical order in which they were entered in the DEFINE operation or as they may have subsequently been modified by the RENAME. See fig. 3b for an alphabetical listing of those labels and of the modules.

```

* * * * *
* FILE FN75 CONTAINS 112 DATA ITEMS FOR 75 RECORDS. *
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* -->? >labels,modules,display *
*
* THE FILE FN75 CONTAINS DATE LABELLED AS FOLLOWS: (A) *
*
* 98 AA 53 AC 33 ADTIT 111 AGE 68 ALC
* 64 ANU 102 APAD 55 APPCR 40 ATD 57 AUDAT
* 51 AUTH 110 C 112 CAGE 92 CAWOL 4 CIT
* 62 CL 91 CLWOP 89 CNHS 36 COLDEG 24 CSDF
* 25 CSDTO 93 CSI 71 DAA 27 DDAY 104 DDED
* 35 DEG 22 DEPT 66 DET 1 DIV 80 DLABR
* 48 DLIM 6 DOB 28 DOGR 30 DOPP 39 DPG
* 76 DPSQ 61 DRPB 50 EDA 31 EOD 32 EODY
* 94 FAB 16 FC 103 FUD 78 GDF 70 GLOC
* 18 GRADE 75 HBPN 44 HC 49 HLIM 82 HWD
* 107 HWWAE 43 INS 23 LC 26 LEGR 88 LWOP
* 101 LWOPSA 67 MC 105 MR 5 NAME 100 NHSA
* 46 NOAC 99 NP 13 NTEDA 63 NTEDP 15 OSC
* 20 PAY 60 PD 74 PF 54 PFC 29 PINFO
* 21 PLANT 52 POS 14 PP 34 PROFS 56 REM
* 45 RET 85 RHWSW 72 RS 109 SALBAS 79 SC
* 8 SCD 69 SCD 108 SCDY 81 SED 41 SEQ
* 3 SFX 106 SK 47 SLIM 58 SON 77 SP
* 59 SPF 38 SPP 2 SS 73 SSNC 9 STAT
* 19 STEP 17 TITLE 90 TLWOP 10 TOA 11 TOD
* 12 TOLA 65 TP 42 VP 86 WAEDWP 84 WAEDWS
* 87 WAEDWT 83 WAEPL 95 WDD 96 WDUD 97 WED
*
* THE FOLLOWING MODULES ARE AVAILABLE: *
* ABRIDGE AGGREGATE ANALYSIS ANNEX
* ARRAY ATTACH BLANKS BROWSE
* CHECKSUM COMPUTE CONCAT CROSSTAB
* DESCRIBE DICTIONARY DISPLAY DISTRIBUTE
* ENCODE EXTRACT FETCH FIT
* GRAPH KWOC MOVE PLAN
* PLOT RANDOM REGRESS RENAME
* REPORT SAVE SCREEN SEARCH
* SEGMENT SEQUENCE SORT STACK
* STATIS STATPLOTS SUMMARY SURVEY
* TALLY TRIM UPDATE UPDATESEQ
* USERS
*
* * * * *

```

Figure 3b. Here we see that when the labels and modules are specifically requested at [A], both are listed in alphabetic order. Note that the order of the input is not fixed for this type of input.

### 3.1 The SEARCH Module

The main retrieval module of the Omnidata system is the SEARCH module. In this module, a data base may be reduced to include only those entries which meet specific criteria. There are many options available to the user in this module.

When the SEARCH module is called in response to the request:

\*TYPE A MODULE NAME AND/OR INSTRUCTIONS --->? >

it responds with

\*TYPE S(ELECT) OR R(EJECT) --->? >

If the response to this questions is:

- a) SELECT or S, the resulting file contains only those records which match the criteria to be specified subsequently; or,
- b) REJECT or R, the resulting file contains those records which *do not match* the criteria to be specified subsequently. In either of the above cases, the criteria are specified in the same way. Regardless of which response was given above, the system requests further information:

\*TYPE LABEL AND VALUE(S) --->? >

1 --->?

If we responded to the above with:

GRADE = 12 (OR GRADE EQ 12)

the SEARCH module responds with

AND

2 --->?

If we now respond by typing END, the system will generate a new file containing only grade 12's, or a file containing everyone but grade 12's, depending upon whether we chose the SELECT or the REJECT mode.

Although this module asks the user to:

\*TYPE S(ELECT) OR R(EJECT) --->? >

there are actually four modes of operation, each of which can be designated in two (or three) ways, as shown below:

SELECT or S or AND

REJECT or R or NAND

SELECT OR or OR

REJECT OR or NOR

The SEARCH module accepts more complex criteria. Thus, if we carry on the dialogue shown in figure 3.1a, the module will look for physicists grade 14 and above who were born in 1925.

Since we did not follow the account number at [A] with the word VERBOSE, we do not get the labels or modules printed. We did, however, follow the account number with MONITOR 100. The result is to be seen at [B], where we are notified (each time 100 records are read) how many hits were found, and how long (in seconds) it took to reach that point. Here we see at [C] that the 3 hits occurred in the first 100 records. This is reasonable since the file is arranged in increasing order of social security numbers. The information supplied by the monitor has many uses which will be pointed out later where appropriate.

In the foregoing, we chose the SELECT mode so the module assumed that we wished to combine a number of characteristics in different data vectors, and it supplied the word AND between each request. If we had responded to the request for the search mode with

SELECT OR (or simply OR)

the module would have assumed that we wanted records in which either one of a number of different data elements were satisfied. Thus, the dialogue shown at the top of figure 3.1b would allow us to have selected from a personnel file all noncitizens (cit=3), plus all nonstatus (TOA GT 2), plus all part-time (TOD=2 OR 3) employees.

The SEARCH module also accepts the response REJECT OR (or simply NOR) in which case the foregoing dialogue would appear as shown in the center of figure 3.1b. Here we would end up with a file which contained none of the above. The file would contain only full-time status employees who are citizens. Other examples of the utility of the OR and NOR options are found in section 5.25.

The module accepts even more complicated criteria. If we typed

TITLE,PHYS OR CHEM OR MATH

we would get not only physicists, chemists, and mathematicians, but also physical chemists, physicians, mathematical statisticians, etc. In short, any job title containing the fragments PHYS or CHEM or MATH would satisfy the search.

Alternatively, if we had typed

TITLE,PHYS AND CHEM

we would get chemical physicists and physical chemists, while

TITLE,PHYS AND ADMIN

would retrieve physical science administrators, administrative physicists, physician administrators, administrative astrophysicists, etc.



```

*****
*      *TYPE P(RINT), S(EARCH) OR E(XIT) --->? >search
*      *TYPE C(URRENT), P(REVIOUS), OR O(RIGINAL) --->? >original
*      *TYPE S(ELECT) OR R(EJECT) --->? >select or
*      *TYPE LABEL AND VALUE(S)
*      1 --->? >CIT = 3
*      OR
*      2 --->? >TOD = 2 OR 3
*      OR
*      3 --->? >TOA GT 2
*      OR
*      4 --->? >END
*      92 HIT(S) WHEN:
*      CIT IS 3
*      OR
*      E TOD IS 2 OR 3
*      OR
*      TOA IS GT 2
*      *TYPE P(RINT), S(EARCH) OR E(XIT) --->? >search
*      *TYPE C(URRENT), P(REVIOUS), OR O(RIGINAL) --->? >original
*      *TYPE S(ELECT) OR R(EJECT) --->? >reject or
*      1 --->? >CIT = 3
*      2 --->? >TOA GT 2
*      3 --->? >TOD = 2 OR 3
*      4 --->? >END
*      408 HIT(S) AFTER REJECTING:
*      CIT IS 3
*      NOR
*      F TOA IS GT 2
*      NOR
*      TOD IS 2 OR 3
*      *TYPE P(RINT), S(EARCH) OR E(XIT) --->? >
*      *TYPE LABEL AND VALUE(S)
*      1 --->? >CIT = 3
*      NAND
*      2 --->? >TOD = 2 OR 3
*      NAND
*      3 --->? >TOA > 2
*      NAND
*      4 --->? >END
*      498 HIT(S) AFTER REJECTING:
*      CIT IS 3
*****

```

Figure 3.1b. Here we have an example of three types of Boolean search in which the criteria are specified in a uniform fashion. At [A] we ask for an OR search to meet the specifications indicated at [B]; at [C] we ask for a NOR search with identical specifications; and at [D] we ask for a NAND search. As there were 500 records in the entire file, the hits shown at [E] and [F] indicate that the first two searches are really complementary.

In the above instructions, the comma which separates the label from its value or values is used to indicate that the characters to the right need only be *contained* in the TITLE field. They need not constitute the entire field. In technical idiom, the comma in this position in the above instructions denotes an unanchored partial string search in the TITLE field. The comma has the same effect even when applied to numeric data fields. Thus, if we had typed CIT,3 in figure 3.1b instead of CIT = 3, any number containing a 3 would have satisfied the search.

In what follows later, the comma is used also to separate input parameters. The discussion of still other uses of the comma in the SEARCH module is deferred until section 5.25.

The SEARCH module is also able to search for exact matches and partial searches anchored to the front or back of words (prefixes and suffixes) or to the front or back end of an arbitrary stream of characters. If an exact match is desired, it is necessary to insert the characters \*E\* between the LABEL and its VALUE. Thus:

DEPARTMENT,\*E\*,MATH

will not match MATHEMATICS or MATHEMATICAL PHYSICS; it will only match MATH. It should be noted, however, that the user need not be concerned where the characters MATH occur in the data field assigned, provided, of course, that the other positions contain blanks. In technical terms, when the SEARCH module is asked for an exact search, (\*E\*), it ignores leading and trailing blanks that may exist in the stored file; it retains interior blanks. When the desired value is a character string containing commas, it is necessary to enclose it between quotes:

ITEM,'CALCULATOR,ROTARY'

In order to select items on the basis of prefixes or suffixes, we employ the asterisk (\*) as follows.

INDEX,ION\*

would match items containing IONIC, IONIZATION, ION, etc., provided the letter I is the first nonblank character in the allotted field.

INDEX,\*ION

would match items containing the words MOTION, NOTION, ION, etc., while

INDEX,' ION '

would match only a data item containing the word ION—the letters ION with a space before the I and after the N. Thus if we were to search a list of titles for those containing the word ION, the above request would not be satisfied if the word ION was followed by a comma or a period as might well be the case in a list of titles. This problem can be solved by

looking for ' ION ', or ' ION,' or ' ION.'. The search instructions for this operation in the OR mode would be:

```
TITLE,' ION '  
TITLE,' ION,'  
TITLE,' ION.'
```

When this module carries out a search, it creates a subfile (a hit file) containing the copies of all of the records that satisfied the search parameters (criteria). The original file is still intact. The user now has three options:

- a) to continue to search either the hit file, which is called the current file, or the original file;
- b) to display all or portions of the data in the original file or in the current file; and,
- c) to exit from the module with either file.

In figure 3.1c we see an example of how it is possible to carry out a search of a file in steps. At [B] we *select* all records where the job title contains the letters PHYS. The resulting file contains 113 records, including among others physical chemists. If we wish to exclude these, we instruct the module at [B] that we wish to search the current (hit file) in the *reject* mode. When we specify at [C] that the title should not contain the letters CHEM, the system discards two records leaving us with a new current file containing 111 records. At this point in the search operation the user has access to three files: the *original* one on which the search was started; the *previous* cut containing 113 records and the *current* cut containing 111 records. If we were now to search the current file (111 records) again, it would become the *previous* file; the file with 113 records, would disappear and there would be a new *current* file. Each time after the first search in a run, the current file becomes the previous file. While the Omnidata system retains only one cut of the file previous to the current one, it is possible to save away, via the SAVE module, as many earlier cuts as one might need.

Another use of the reject mode occurs when it is desired to extract data on all compounds containing H, D, N, and O except those containing both H and C. The instructions for the *select* mode would be

```
ATOM,H AND D AND N AND O
```

followed by another search on the current file in the *reject* mode for

```
ATOM,H AND C
```

Whether we select first or reject first is immaterial as far as the result is concerned. The order may influence the cost of the search depending upon the relative proportions of hydrocarbons in the file.

```

*****
* PLEASE ENTER ACCOUNT NUMBER-->? >a534, monitor 100 *
* TYPE PASSWORD --->? >1112 *
* WHICH DATA BASE DO YOU WANT -->? >fndemo *
* GOOD MORNING, WELCOME TO OMIDATA *
* * *NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH *
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY *
* FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS. *
*
* TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* --->? >search *
*
* TYPE S(ELECT) OR R(EJECT) --->? >select *
* TYPE LABEL AND VALUE(S) *
* 1 --->? >title, phys *
* AND *
* 2 --->? >end *
* 100 RECORDS READ 33 HITS 6.0688 *
* 200 RECORDS READ 44 HITS 6.6204 *
* 300 RECORDS READ 70 HITS 7.1818 *
* 400 RECORDS READ 95 HITS 7.7494 *
* 500 RECORDS READ 113 HITS 8.28 *
* 113 HIT(S) WHEN: *
* TITLE IS PHYS *
* TYPE P(RINT), S(EARCH) OR E(XIT) --->? >search *
* TYPE C(URRENT), P(REVIOUS), OR O(RIGINAL) --->? > current *
* TYPE S(ELECT) OR R(EJECT) --->? >reject *
* TYPE LABEL AND VALUE(S) *
* 1 --->? >TITLE, CHEM *
* NAND *
* 2 --->? >end *
* 100 RECORDS READ 99HITS 9.0936 *
* 111 HIT(S) AFTER REJECTING: *
* TITLE IS CHEM *
* TYPE P(RINT), S(EARCH) OR E(XIT) --->? >stop *
* PROGRAM STOPPED. *
*****

```

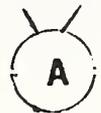


Figure 3.1c. Here we see how the use of the SELECT search mode (for TITLE, PHYS) at [B], followed by the REJECT search mode (for TITLE, CHEM) at [D], deletes two physical chemists or chemical physicists (or one of each) from a file of persons whose job title contains the fragment PHYS. The file may still contain physicians, physical therapists, etc. These could also be eliminated in a similar fashion. The report at [C] results from the monitor instruction at [A]. The distribution of hits among the records in a file is useful information in certain well-structured files. In any case it tells the user that the program is processing the data.

In the case of searching numeric data vectors, the search module accepts the symbols < or LT for *less than*, > or GT for *greater than*, and = or EQ for *equal to*. The search can also be instructed to look for a range of values. Thus:

```
YOB,>,1939
AND
YOB,<,1951
```

would result in records in which the years of birth (YOB) would fall in the range 1940 to 1950 inclusive. An alternate way of achieving the same result is via the instruction:

```
YOB,CIRCA,1940-1950
```

Other features of the SEARCH module are discussed in section 5.25. The more important of these provide for:

- a) ignoring specific characters in the file during the searching operation;
- b) locating only the first record or the first block of records that satisfies a given criterion;
- c) limiting the search to a specific block of records;
- d) selecting records in which the entry (value) for one vector (attribute) equals that for another vector;
- e) locating records having missing (blank) data entries in certain data vectors; and
- f) taking into account the order and distance between string fragments or words in a data item.

*3.1.1 The EXIT and PRINT Options in the SEARCH Module.* Each time the SEARCH module finishes its operation, it types:

```
*TYPE P(RINT, S(EARCH), OR E(XIT)--->? >
```

Regardless of the reply, it is necessary to inform the module which version of the file one wishes to be available for the next operation. That information is supplied in reply to the question

```
*TYPE C(URRENT), P(REVIOUS) OR O(RIGINAL)--->? >
```

If this question is to be answered after the first pass through the SEARCH module there is no previous file—there is only the original and current file. If we EXIT with this file and SAVE it, perform another search on it, and save that (current) file, we would now have the original and two partial files saved away for future use.

The PRINT provides the user with an opportunity to print out information from a file while still under the control of the SEARCH module. The facilities under the PRINT option are similar to, but are not as extensive as, those available in the DISPLAY module which is discussed in the next section.

Each time the SEARCH module finishes its search, it types:

```
*TYPE P(RINT), S(EARCH), OR E(XIT)--->? >
```

On a response of PRINT or P the SEARCH module requests:

```
*TYPE LABELS OF ITEMS TO BE PRINTED OR ALL  
--->? >
```

ALL, in this case, means all data items (not all records) and even then for only a single record at a time. Thus, in figure 3.1d there are displayed the data items for the first record in an inventory file. Each data item is labeled in accord with the way the file was defined. If the resulting file contains only a handful of records, a response of ALL each time the module asks \*MORE--->? > is an efficient way of getting at the information. This is especially true when there are a large number of data items per record. A printout such as is shown in figure 3.1d is useful as a reminder of how the data items are labeled. It is often informative as regard to the way the information is entered in the file (alphabetic, numeric, or mixed).

At this point, we can also answer the question \*MORE--->? > with a number, or the words YES or NO. If we answer YES, we get the same information (all data items) for the next record. If we answer with a number, n, we get the same information in the same format for the next n records. Until we answer NO to the question (MORE--->? > we are locked into the ALL option.

Whenever we answer this question with NO, we are given an opportunity to specify again which data items are to be displayed. Figure 3.1e shows how the PRINT feature responds when instructed to print out the NOMENCLATURE data item in the inventory file. It starts with the second record because the first had already been printed in the previous instruction. There is no provision in the PRINT option to back up in the file. Such a provision is, however, built into the DISPLAY module.

When we answer NO to the question \*MORE--->? >, we break the format cycle but not the PRINT cycle, as is clear from figure 3.1e. In order to get out of the PRINT cycle we must type END and the computer response at [D] shows that we are still in the SEARCH module from which we now wish to EXIT.

```

*****
*
*      *TYPE LABELS OF ITEMS TO BE PRINTED OR ALL
*
*      -->? >all
*
*
*
*      NBS -- 005202
*      T/C -- X06
*      DIV -- 122
*      SEC -- 00
*      ORG -- 12200
*      PROJ -- 000
*      DOLLAR -- 0000441
*      SCH -- 3
*      LIFE -- 12
*      ACQ -- 0660
*      ITEM --
*      MFR -- R429
*      MODEL -- ELECTRIC
*      SERIAL -- E2338496
*      VOUCHER --
*      PURCHASE -- 34238
*      ACCUM -- 000000000
*      MONTHCH -- 000000024
*      CLASS -- 3
*      NOMENCLATURE -- TYPEWRITER ELECTRIC
*      MANUFACTURER -- REMINGTON RAND INC.
*      AVAILABILITY --
*      SPECS --
*      LASTACT -- 0171
*      LASTVOUCHER -- 8276771279
*      SEG --
*      FLAG --
*      LASTT/C --
*      FLAG2 --
*      ITEMSEQCODE -- 000000
*      BLDG-RM -- B35      101
*      OFF-SITE --
*
*
*
*      *MORE --->? >no
*
*****

```

Figure 3.1d. Here we see a display of the data in the first record of an equipment inventory file produced by the PRINT option in SEARCH. The ALL response at [A] mean all data items. An answer of YES at [B] will produce the same information for the next record. A response of NO allows the user to specify particular data items to be displayed as seen in figure 3.1e et seq.

```

*****
*
*      *TYPE LABELS OF ITEMS TO BE PRINTED OR ALL
*
*  --> ? >nomenclature
*
*  REC      NOMENCLATURE
*
*
*  2      TYPEWRITER ELECTRIC, IBM STANDARD
*  3      CALCULATOR ROTARY
*  4      TYPEWRITER MANUAL
*  5      CALCULATOR ROTARY
*  6      TYPEWRITER MANUAL
*  7      TYPEWRITER MANUAL
*  8      CALCULATOR ROTARY
*  9      TYPEWRITER MANUAL
*  10     TYPEWRITER MANUAL
*
*  MORE -->? >20
*
*  11     ADDING MACHINE
*  12     TYPEWRITER MANUAL
*  13     CALCULATOR ROTARY
*  14     TYPEWRITER MANUAL
*  15     CALCULATOR ROTARY
*  16     TYPEWRITER MANUAL
*  17     CALCULATOR ROTARY
*  18     TYPEWRITER MANUAL
*  19     ADDING MACHINE
*  20     CALCULATOR ROTARY
*  21     TYPEWRITER MANUAL
*  22     TYPEWRITER MANUAL
*  23     CALCULATOR ROTARY
*  24     CALCULATOR ROTARY
*
*-----*
*  30     CALCULATOR MILLIONAIRE
*  31     CALCULATOR ROTARY
*
*
*  *MORE -->? >no
*
*  *TYPE LABELS OF ITEMS TO BE PRINTED OR ALL
*
*  --> ? >end
*
*
*  *TYPE P(RINT), S(EARCH) OR E(XIT) -->? >EXIT
*
*****

```

Figure 3.1e. When specific data items are requested as at [A] under the PRINT option of SEARCH, the module displays the requested item(s) for the next 10 records. A response of YES at [B] would result in output for 10 more records, while a number produces that many records. A response of NO at [C] allows us to specify a different set of data items, but starting with the next record. As the PRINT option in SEARCH does not have a back-up provision, it is necessary to return to the SEARCH module by typing END as at [D] if we wished to start another printout with the first record.

```

*****
* 113 HIT(S) WHEN:
* TITLE IS PHYS
* *TYPE P(RINT), S(EARCH) OR E(XIT) --->? > search
* *TYPE C(URRENT), P(REVIOUS), OR O(RIGINAL) --->? >current
*
* *TYPE S(ELECT) OR R(EJECT) --->? >reject
* *TYPE LABEL AND VALUE(S)
* 1 --->? >title, chem
* NAND
* 2 --->? >end
* 111 HIT(S) AFTER REJECTING:
* TITLE IS CHEM
* *TYPE P(RINT), S(EARCH) OR E(XIT) --->? | >print
* *TYPE C(URRENT), P(REVIOUS), OR O(RIGINAL) --->? >current
* *TYPE LABELS OF ITEMS TO BE PRINTED OR ALL
* ---> ? >div, yob, title ----- A
*
* D YO TITLE ----- B
* 2 15 SUPERVISORY PHYSICIST
* 2 22 PHYSICAL SCIENCE
* 3 37 PHYSICIST (SOLID)
* 2 25 SUPERVISORY PHYSICIST
* 3 29 PHYSICIST (GENERAL)
* 2 25 PHYSICIST (PHYSICS)
* 2 19 PHYSICAL SCIENCE
* 2 38 PHYSICIST (NUCLEAR)
* 4 21 PHYSICIST (SOLID)
* 2 25 PHYSICAL SCIENCE
* *MORE --->? >yes
* 4 20 PHYSICAL SCIENCEA
* 2 52 PHYSICAL SCIENCE
* 2 22 PHYSICIST (NUCLEAR)
*-----
* 4 44 PHYSICIST (HEAT)
* 2 29 PHYSICIST (GENERAL)
* 3 28 RESEARCH PHYSICIST
* *MORE --->? >no
* *TYPE P(RINT), S(EARCH) OR E(XIT) --->? >exit
* *TYPE C(URRENT), P(REVIOUS), OR O(RIGINAL) --->? >current
*
*****

```

Figure 3.1f. A printout of the first 20 records of the file resulting from the search strategy illustrated in figure 3.1c. At [A] we ask to see three data items (DIV, YOB, TITLE). These are shown, 10 at a time, under the column headings at [B] where the DIV appears as D and YOB appears as YO. These column headings have been truncated because the formatting is controlled by the width of the data field rather than the width of the labels associated with them. The title field has been truncated to extend over two words for demonstration purposes.

### 3.2 The DISPLAY Module

This module provides a quick way of looking at information stored in a file. While it is intended primarily for just looking at the information, it can also be used for reporting data when the formatting is not crucial. This module displays data in two major modes. In the first mode (see figs. 3.2a and 3.2b), all data items are displayed; each of the data points is labeled (tagged). In the second mode, where only selected data items are displayed, they are not tagged since the labels are displayed at the top of the page.

```
*****
*
*          *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*  --->? >ALL
*  ID -- 66-IN-DE-A5
*  CCODE -- IN
*  CINST -- DE
*  PROGC -- A5
*  TITLE -- COMPILATION OF CRITICAL TABLES OF SRD
*  OF DEFECT PROPERTIES ON-METALLIC CRYSTALS
*  OF DEFECT PROPERTIES ON-METALLIC CRYSTALS
*  NAME -- S.C. JAIN
*  ADD -- DEPT. OF PHYSICS, INDIAN INSTITUTE
*  OF TECHNOLOGY NEW DELHI
*  COUNTRY -- INDIA
*  NBSNAME --#####
*  DIV -- 151
*  AMOUNT -#####
*  YB -- 68
*  DB -- 680313
*  YE -- 72
*  DE -- 720729
*  AM72 -- 0
*  REM -- SUPPLEMENT #1 and #2
*
*          *MORE --->? >NO
*****
```

Figure 3.2a. A display of one record of data in a project file showing how the data are labeled when ALL data items are requested.

In figures 3.2a and 3.2b, the response ALL refers not to all of the records but to all of the defined data items. In this mode, information is printed for only the first record. After the first record, the user can specify how many records should be displayed in the next cycle.

When the user responds with specific labels instead of ALL, this module prints out information for five records as shown in figure 3.2c. After this initial information, the user can specify how many records should be displayed in the next cycle. Two features of this printout should be noted. First, the data are displayed in the order in which the labels are entered and, secondly, the column headings are truncated to the defined width of the data field rather than the length of the label. The latter shortcoming can be overcome by using the REPORT module.

When the DISPLAY module asks for \*MORE --->? >, the system ordinarily expects either YES, NO, or a number. If the response is YES, data is displayed for five more records. The consequences of the other responses should be self-evident.

Since it is often useful to be able to look *at* a specific record in a file or look *for* a specific record, this module will also respond to such commands as

SKIP TO 100 or BACK TO 50.

Thus, after seeing the first record, the user can look at the 100th record and then back up to the 50th. This facility allows one to zero in on a particular record in a systematically organized file much faster and at much less cost than if one were to search through the file. It is worth remembering at this point that if the global command BACKWARD had been given earlier this module would display or otherwise operate from the end of the file instead of the beginning.

Other examples of the use of the DISPLAY module are shown in section 5.10.

```

*****
*                                     *
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL                       *
*      --> ? >all                                                         *
*                                     *
*      RECORD NUMBER 1                                                    *
*      DIV -- 3                                                            *
*      SS  --                                           *
*      SEX -- 2                                                            *
*      CIT --                                           *
*      NAME --                                           *
*      DOB -- 121737                                                       *
*      YOB -- 37                                                            *
*      SCD -- 042568                                                       *
*      STAT -- 1                                                            *
*      TOA -- 1                                                            *
*      TOD -- 2                                                            *
*      TOLA -- 2222                                                         *
*      NTEDA -- 000000                                                      *
*      PP  -- GS                                                            *
*      OSC -- 03120                                                         *
*      FC  --                                           *
*      TITLE -- CLERK STENOGRAPHER                                         *
*      GRADE -- 04                                                         *
*      STEP -- 06                                                         *
*      PAY  -- 08027                                                         *
*      PLANT --                                           *
*      DEPT -- 31402                                                         *
*      LC  -- 240630031                                                      *
*      CSDF -- 040757                                                       *
*      CSDTO -- 040660                                                       *
*      LEGR -- 24                                                            *
*      DDAY -- 12217                                                         *
*      DOGR -- 112972                                                       *
*      PINFO -- 121001                                                       *
*      DOPP -- 040573                                                       *
*      EOD  -- 112972                                                       *
*      EDDY --                                           *
*      ADTIT --                                           *
*      PROFS -- 5                                                            *
*      DEG  --                                           *
*      COLDEG --                                           *
*      YR  --                                           *
*      SPP  --                                           *
*      DPG  -- 112972                                                       *
*      ATD  --
*****

```

```

SEQ -- 5207550261
VP  -- 12
INS -- 1
HC  -- 00
RET -- 1
NOAC -- 89400
SLIM -- 00000
DLIM -- 000
HLIM -- 0000
EDA  -- 062473
AUTH --
POS  -- 30246000
AC  -- CW13
PFC  -- 0
APPOR -- 00
REM  -- 111153
AUDAT -- 062473
SON  -- 1703
SPF  -- 22222
PD   --
DRPB --
CL  -- 105
NTEDP -- 000000
ANU  -- 2
TP   -- 2
DET  -- 2
MC  -- 7
ALC  --
SCD  --
GLOC -- 0010
DAA  --
RS   -- 0
MR  -- 6
SK  -- 8
HWWAE -- 0000
SCDY -- 68
SALBAS -- PA
C   -- 00

```

Figure 3.2b. A display of part of the information in the first record of a personnel file showing how the data items are identified when ALL data items are requested. This feature is also available under the PRINT option in the SEARCH module.

```

* * * * *
*
*          *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*
*  --> ? >rec, nbspart
*  REC   REC   N
*  ----  -
*  1      5001   O
*  2      5002   T
*  3      5003   T
*  4      5004   T
*  5      5005   T
*
*      *MORE -->? >skip to 100
*      *MORE -->? >yes
*  100    5337   A
*  101    5350   O
*  102    5351   T
*  103    5352   A
*  104    5353   T
*  105    5364   O
*  106    5365   A
*  107    5366   T
*  108    5367   A
*  109    5368   T
*
*      *MORE -->? >back to 20
*      *MORE -->? >3
*  20     5032   T
*  21     5033   T
*  22     5034   T
*
*      *MORE -->? >skip to 150
*      *MORE -->? >yes
*  150    7027   A
*  151    7028   A
*  152    7029   A
*  153    7030   A
*  154    7031   A
*  155    7032   A
*  156    7033   A
*  157    7034   T
*  158    7035   A
*  159    7036   A
*
*      *MORE -->? >stop
*
*  PROGRAM STOPPED.
*
* * * * *

```

Figure 3.2c. Here we see that when specific data items are requested for display the module produces the information for five records. After these are printed, the user has options for skipping around in the file as shown.

```

* * * * *
*
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --> ? x-crd, y-crd, x1-crd, y1-crd, x2-crd, y2-crd, &
* ? zip, ext, bldg
*
*      X-C      Y-C      X1-C      Y1-C      X2-CR      Y2-CR      ZIP      EXT      BLDG
*      -----
*
*      3      36      32      361      326      3610      21760      3697      TECH
*      0      29      60      293      609      2933      21793      3101      S&P
*      999    999    9999   9999   99999   99999   19809      2153      CHEM
*      7      16      77      165      773      1650      20767      3405      ADMIN
*      11     14      113     146      1138     1468      20850      5677      POLY
*      15     13      155     132     1557     1320      20902      2752      POLY
*      999    999    9999   9999   99999   99999   94087      3461      TECH
*      17     12      178     129     1781     1298      20705      2625      RADP
*      9      16      94      163     942      1633      20760      2634      REACT
*      999    999    9999   9999   99999   99999   20784      2121      MET
*
*      *MORE -->? no
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --> ? x-crd, y-crd, x1-crd, y1-crd, x2-crd, y2-crd, &
* zip, ext, bldg.?
* zip, ext, room, bldg
*
*      X-C      Y-C      X1-C      Y1-C      X2-CR      Y2-CR      ZIP      EXT      ROOM
*
*      BLDG
*      3      36      32      361      326      3610      21760      3697      A 109
*      TECH
*      6      29      60      293      609      2933      21793      3101      B 141
*      S&P
*      999    999    9999   9999   99999   99999   19809      2153      A 110
*      CHEM
*      7      16      77      165      773      1650      20767      3405      E 120
*      ADMIN
*      11     14      113     146      1138     1468      20850      5677      A 155
*      POLY
*      15     13      155     132     1557     1320      20902      2752      A 125
*      POLY
*      999    999    9999   9999   99999   99999   94087      3461      B 47
*      TECH
*      17     12      178     129     1781     1298      20705      2625      C 107
*      RADP
*      9      16      94      163     942      1633      20760      3634      A 159
*      REACT
*      999    999    9999   9999   99999   99999   20784      2121      A 47
*      MET
*
*      *MORE -->? no
*
* * * * *

```

Figure 3.2d. Here we see how the DISPLAY module format varies when the requested data items fit on a single line and when they do not.

### 3.3 The REPORT Module

As the object of operating on a computerized data file is to obtain information in a useful and convenient format, a versatile report generator is a very important component of a data management system. Thus, the REPORT module has been designed to provide the user with considerable flexibility in arranging the output to meet varied requirements.

The REPORT module operates in a number of modes and various modifications thereof as follows:

- 1) A *labeled* mode in which each data point in the report is preceded by its name taken from the labels associated with the data file, or as modified in the RENAME module. See fig. 3.3a.
- 2) An *unlabeled* mode where each data item is printed without any identification except for optional column headings as shown in figure 3.3b.
- 3) A mixed mode in which certain items are exceptions to the above rule. Thus, in figure 3.3b, where we chose to print out the name of the project leader, there was no need to precede it with NAME = , as it would be obvious that the information printed was a person's name.

An examination of figure 3.2a will show a number of features. The user can specify the number of characters per line, the number of lines per "page", and any number of lines of title or footnotes to be printed on each page. Since this report consists of single line entries, the program asks for column headings. At this stage there are three options:

- a) no column headings as shown in figure 3.3a;
- b) column headings taken from the labels and truncated if necessary (see fig. 3.3b); and
- c) ad hoc column headings supplied from the keyboard.

In figure 3.3b we show the conversation between the system and the user which allows one to modify certain aspects of the column headings for a report prepared in the unlabeled mode. In section 5.23 we show how the report module allows the user to space out his data columns and provide column headings of a more complex structure.

```

*****
*
*      *TYPE A MODULE NAME OR INSTRUCTIONS
*      ---> ? > report
*      *TYPE MODE--L(ABELLED) OR UNL(ABELLED)-->? unl
*      *TYPE CHARACTERS PER LINE --->? 80
*      *TYPE LINES FOR OUTPUT
*      LINE 1 ? restart
*
*      *TYPE MODE--L(ABELLED) OR UNL(ABELLED)-->? 1
*      *TYPE CHARACTERS PER LINE --->? 80
*      *TYPE LINES FOR OUTPUT
*      LINE 1 ? ss , dob, eod, scd
*      LINE 2 ? end
*
*      *TYPE TITLE LINES OR N(ONE) --->
*      ? 1, test of labelled mode
*      ? end
*
*      *ANY COL HEADS--TYPE L(ABELS), N(ONE) OR 4 HEADINGS--->
*      ? n
*
*      *TYPE FOOTNOTE LINES OR N(ONE) --->
*      ? 1, these data are ficticious
*      ? end
*
*      *DO YOU WANT TO POSITION PAPER AFTER EACH PAGE--
*      TYPE YES OR NO --->? yes
*
*-----
*
*      TEST OF LABELLED MODE
*
*-----
*
*      SS = 797049629   DOB= 110629   EOD= 063065   SCD= 110663
*      SS = 168596476   DOB= 102119   EOD= 102171   SCD= 100672
*      SS = 777303254   DOB= 110927   EOD= 092462   SCD= 041061
*      SS = 894247697   DOB= 123138   EOD= 145763   SCD= 820735
*      SS = 507003098   DOB= 555852   EOD= 111075   SCD= 070171
*      SS = 123098571   DOB= 145280   EOD= 230676   SCD= 070170
*      SS = 444202763   DOB= 060452   EOD= 028600   SCD= 061668
*      SS = 628358478   DOB= 101144   EOD= 090170   SCD= 031443
*      SS = 222422222   DOB= 082319   EOD= 030265   SCD= 090170
*      SS = 281593322   DOB= 082319   EOD= 030265   SCD= 031752
*      SS = 596987064   DOB= 082319   EOD= 120351   SCD= 082342
*      SS = 541782606   DOB= 030926   EOD= 022745   SCD= 022745
*
*      *NOTE: THESE DATA ARE FICTICIOUS
*****

```

Figure 3.3a. A record of the use of the REPORT module to produce a labeled report. See the text for a discussion of the numbered features shown here and on the continuation following. The title is shifted to the right because we specified a width of 80 characters.

The characteristics of the REPORT module illustrated in figure 3.3a are as follows:

- 1) Here we asked for an unlabeled report and then changed our mind by responding with the word RESTART instead of supplying the information desired on the first line of the report. RESTART can be used in response to any input request from this module.
- 2) Print lines can be as long as 132 characters provided that the system was set to that figure previously as discussed in section 1.6.2.
- 3) The module types the line numbers automatically until the word END appears.
- 4) The other responses that are accepted at this point are RETURN and DUPLICATE. These are discussed below.
- 5) Here the user supplies the line numbers so that a repetition of a line previously defined updates that instruction. Furthermore, the omission of a line in the sequences produces a blank line.
- 6) Since we specified only one line of output, the module allows for column headings which are not really required in the labeled mode. Hence the answer is N for none.
- 7) Any number of footnote lines can be entered. They appear on each page of the report. The word NOTE is supplied by the module on the first line only. Subsequent lines are indented seven spaces.
- 8) If the response is YES, the printout stops after each page to allow for positioning of the paper. The printing is resumed after the user types any character and returns the carriage. When operating in the batch mode, the response to this question must be NO.

Other features of this module are shown in figure 3.3b as follows:

- 1) Here we ask for a report in the unlabeled (unl) mode.
- 2) When the instructions are ended with RETURN instead of END, the module allows for generating another type of report from the same file without recalling the REPORT module. Copies of a report can be ordered at this point by responding with the word DUPLICATE, n. Here n is the number of copies desired. If a number does not follow the word DUPLICATE, the module prepares two copies.



671444920 020655 BEHUSIUU, DAFOOIO K. [MISS]

\*NOTE: THESE DATE ARE FICTICIOUS

-----  
\*TYPE MODE-L(ABELLED) OR UNL(ABELLED)- -->? 1  
\*TYPE CHARACTERS PER LINE -->? 60  
\*TYPE LINES FOR OUTPUT  
LINE 1 ? ss ,dob, name, unl  
LINE 2 ? repeat  
\*TYPE TITLE LINES OR N(ONE) --->  
? 1, test of mixed mode  
? 1, report number"er one  
? 3, mixed mode  
? END  
\*ANY COL HEADS- -TYPE L(ABLES), N(ONE) OR 3 HEADINGS --->  
? n  
\*TYPE FOOTNOTE LINES OR NONE) --->  
? 1, these data are ficticious  
? end  
\*DO YOU WANT TO POSITION PAPER AFTER EACH PAGE- -  
TYPE YES OR NO --->? yes  
-----

REPORT NUMBER ONE  
MIXED MODE  
-----

SS = 218161630	DOB= 011009	GIGABO, SIEOQ X.	[MR. ]
SS = 805151288	DOB= 122713	OOAIF, OEET Q.	[MR. ]
SS = 970309286	DOB= 063014	EAAILITA, JOKEVO E	[MRS.]
SS = 507003098	DOB= 011114	XIQB, EJUZUKE EZEUA	[MR. ]
SS = 506987064	DOB= 082319	UUPAFU, TINEKEAI	[MR. ]
SS = 168596476	DOB= 102119	DEPUCT JJKLE E	[MIS ]
SS = 007655895	DOB= 013121	FIVTRA SEK Q.	[MRS.]
SS = 123788999	DOB= 060452	WOCOOAL, IOWUX	[MR.]
SS = 666175624	DOB= 012553	RUBUAAA, EBOBINI	[MR. ]
SS = 455227492	DOB= 010555	ZEBEPIM, ESIRAJA E.	[MIS ]
SS = 671444920	DOB= 020655	BEHUSIUU, DAFOUOIO	[MIS ]

\*NOTE: THESE DATA ARE FICTICIOUS

Figure 3.3b (concluded). Here we ask for a second report to illustrate the use of the mixed mode of presentation (some data labeled and others not).

- 3) Here we tell the module to use SS# DOB, NAME as column headings instead of the labels associated with the data base. These are printed out in their normal position relative to a line of the required report to make sure that the positioning is satisfactory. Had we formatted the report to cover more than a single line, the option to provide labels would be bypassed.
- 4) The NO response allows the user to shift the positions of the labels or to change one or more of them completely. Here we chose to shift them a little to the right and change SS to SS#.
- 5) Here we indicate satisfaction with the format and the module carries on in the normal manner.
- 6) When this report runs to more than one page, the pages are numbered starting on page 2.
- 7) Since we ended the instructions for the first report with RETURN instead of END, the module now asks for instructions for a second report. This time we ask for a report in the labeled (L) mode with the exception noted below.
- 8) Here we ask that the NAME field be printed without a label (unl for unlabeled).

In figure 3.3c, we show features provided to space out headings and to modify them. Provisions for inserting spaces between the columns in the report proper are illustrated in section 5.24.

- 1) Here we give instructions to prepare the report with 20 lines per page. Had we omitted the second number (20) this module would supply as many lines per page as needed to make the output proportional to a 6x9 print area. Due allowance is made for lines in the title, footnotes, and column headings in formatting the page.
- 2) Since seven data items were requested in the report this module asks for seven column headings, which we supply at [3].
- 4) Here we would prefer to have ID centered over the data and wish to have the last column headed by AMT instead of AMOUNT. The instructions to achieve these changes are shown in item four.
- 5) The user supplies the line number followed by the changes desired. Note that each change is indicated separately and especially the spaces following the ID label. Complications resulting from omitting these spaces are shown in the next figure.

- 6) Here we are shown how the revised heading will look relative to the data columns. The module allows us to make changes until it receives the response YES, after which the user can specify footnotes and instructions on positioning of the paper between pages.

Had we responded at [3] with the word LABELS, this module would have used the labels indicated at input [2] and centered them automatically over the columns (suitably truncated, if necessary).

There are other features in this module. Examples of their use and operation has been deferred until section 5.23. Among the more important of these are provisions for:

- a) interrupting the printing of a report;
- b) indicating labels to be excepted from a report when the list of labels is much longer than the list of exceptions;
- c) suppressing the paging of a report;
- d) inserting spaces or literal expressions between columns of a report;  
and
- e) writing the report on a file in addition to or instead of the terminal.

\* \* \* N O T E S \* \* \*

```

* * * * *
*
*   *WHICH MODULE DO YOU WANT --> ? REPORT
*
*   *TYPE MODE--L(ABELLED) OR UNL(ABELLED)-->? unl
*   *TYPE DIMENSIONS OF LINE AND PAGE --->? 80,20
*   *TYPE LINES FOR OUTPUT
*   LINE 1 ? id, div, yb, ye, ccode, cinst, amount
*   LINE 2 ? end
*   *TYPE TITLE LINES OR N(ONE) --->
*   ? none
*   ANY COL HEADS- -TYPE L(ABLES), N(ONE) OR 7 HEADINGS --->
*   ? id, div, yb, year, ccode, cinst, amount
*   ? end
*   ID          DIV      YB      YE      CCO  CIN  AMOUNT
*   18-IN-PA-A5  240      63      73      IN   PA   71.5
*   *OK- -TYPE YES OR NO --->? NO
*   *TYPE LINE NO., OLD HEAD, NEW HEAD --->
*   ? 1, amount, amt
*   ? 1, 'id          '      id no.'
*   ? end
*   ID NO.      DIV      YB      YE      CCO  CIN  AMT
*   18-IN-PA-A5  240      63      73      IN   PA   71.6
*   *DO- -TYPE YES OR NO --->? yes
*   *TYPE FOOTNOTE LINES OR N(ONE) --->
*   ? none
*
*   *DO YOU WANT TO POSITION PAPER AFTER EACH PAGE- -
*
*   TYPE YES OR NO --->? yes
*
* 3206 - - - - -
*
*   ID. NO.      DIV      YB      YE      CCO  CIN  AMT
*   =====
*   18-IN-PA-A5  240      63      73      IN   PA   71.6
*   50-IN-KA-A8  313      65      72      IN   KA   21.5
*   66-IN-DE-A5  151      68      72      IN   DE   37.6
*
* * * * *

```

Figure 3.3c. An example of the provision in the REPORT module to modify the column headings. See the text for a discussion of the numbered items. Note that the headings YEAR, CCODE, and CINST were truncated in the column headings to fit the defined column width of these data items. Had we responded at [3] with the word LABELS, this module would have centered the normal labels shown on line two over the columns (suitably truncated, if necessary).

```

*****
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -->? report
*      *TYPE MODE-L(ABELLED) OR UNL(ABELLED)- -->? unl
*      *TYPE DIMENSIONS OF LINE AND PAGE- -->? 80
*      *TYPE LINES FOR OUTPUT
*      LINE 1 ? id,div,yb,ye,ccode,cinst,amount
*      LINE 2 ? end
*      *TYPE TITLE LINES OR N(ONE) - -->
*      ? none
*      *ANY COL HEADS- - TYPE L(ABELS),N(ONE) OR 7 HEADINGS- -->
*      ? labels
*      ? end
*      ID          DIV    YB    YE    CCO CIN  AMOUNT
*      18-IN-PA-A5  240    63    73    IN  PA  71.5
*      *OK- -TYPE YES OR NO- -->? NO
*      *TYPE LINE NO., OLD HEAD, NEW HEAD- -->
*      ? 1,amo,amt
*      ? 1,id,' id no.'
*      ? end
*      ID          DIV    YB    YE    CCO CIN  AMOUNT
*      18-IN-PA-A5  240    63    73    IN  PA  71.5
*      *OK - -TYPE YES OR NO- -->? RESTART
*      *TYPE MODE- -LABELLED OR UNLABELLED- -->? ul
*      *TYPE DIMENSIONS OF LINE AND PAGE- -->? 80
*      *TYPE LINES FOR OUTPUT
*      LINE 1 ? id,div,yb,ye,ccode,cinst,amount
*      LINE 2 ? end
*      *TYPE TITLE LINES OR N(ONE)- -->
*      ? none
*      *ANY COL HEADS- -TYPE L(ABELS), N(ONE) OR 7 HEADINGS- -->
*      ? labels
*      ? end
*      ID          DIV    YB    YE    CCO CIN  AMOUNT
*      18-IN-PA-A5  240    63    73    IN  PA  71.5
*      *OK- -TYPE YES OR NO- -->? NO
*      *TYPE LINE NO., OLD HEAD, NEW HEAD- -->
*      ? 1,amount,amt
*      ? 1,id,' id no.'
*      ? end
*      ID          DIV    YB    YE    CCO CIN  AMT
*      18-IN-PA-A5  240    63    73    IN  PA  71.5
*      *OK- -TYPE YES OR NO- -->? restart
*****

```

Diagram annotations: A circled '1' has arrows pointing to the 'id no.' label and the 'amt' column header in the second table. A circled '2' has arrows pointing to the 'id no.' label and the 'AMT' column header in the third table.

Figure 3.3d. A record of two unsuccessful attempts to modify and shift column headings. The correct instructions for this operation are shown in item 4 of the previous figure.

```

* * * * *
* *TYPE MODE--L(ABELED) OR UNL(ABELED) ---->? >unl
* *TYPE CHARACTERS PER LINE ---->? >60
* *TYPE LINES FOR OUTPUT
* LINE 1 ? ss#,dob,eod,scd
* LINE 2 ? end
* *TYPE TITLE LINES OR N(ONE) ---->
* ? 1,we will put in col heads
* ? end
* *ANY COL HEADS--TYPE L(ABELS),N(ONE) OR 4 HEADINGS---->
* ? social,date,entered,service
* ? security,of,on,comp
* ? num,birth,duty,date
* ? end
* SOCIAL      DATE      ENTERED  SERVICE
* SECURITY    OF        ON        COMP
* NUM        BIRTH    DUTY      DATE
* 410064386  101928  061266  061662
* *OK--TYPE YES OR NO---->? >yes
* *TYPE FOOTNOTE LINES OR N(ONE)---->
* ? n
* PLEASE POSITION PAPER
* ? ok
* -----
*
*
* WE WILL PUT IN COL HEADS
* -----
*
* SOCIAL      DATE      ENTERED  SERVICE
* SECURITY    OF        ON        COMP
* NUM        BIRTH    DUTY      DATE
* =====
* 410064386  101928  061266  061662
* 562639893  010722  062765  102558
* 748893016  120742  071369  071369
*
* -----
*
*
* -2-
* WE WILL PUT IN COL HEADS (CONT)
* -----
*
* SOCIAL      DATE      ENTERED  SERVICE
* SECURITY    OF        ON        COMP
* NUM        BIRTH    DUTY      DATE
* =====
* * * * *

```

Figure 3.3e. Here we see how this module accepts instructions at [B] for multiple line column headings, how it places these over the appropriate columns in the report at [C], and how it handles heading lines and page numbers on the second and succeeding pages. The title is centered over an 80 character line as was requested at [A].

### 3.4 The UPDATE Module

This module provides for correcting or modifying the individual data items in an existing Omnidata file. It does not delete or insert or move entire records. Those operations are available in other modules. Many files contain a unique identifier for each record — such as social security number, part number, reference number, etc. — which in principle is sufficient to locate the record to be modified. Because it is so easy to transpose digits in such numbers, Omnidata makes provision for identifying records by as many as 10 different data items. In addition, when the update is performed on-line, the system can type out one or more data items for more positive verification before the update is accepted.

Whether in batch or demand mode, this module accumulates all of the update instructions and writes them on a scratch file before carrying them out. The following are some typical instructions typed in response to the request:

```
*TYPE IDENTIFICATION AND UPDATE INSTRUCTIONS --->? >
? FIND NAME = SMITH, YOB = 36, VERIFY SS#,
  LET SCD = 64, GRADE = 9, STEP = 2
? FIND SS# = 1273548, LET DEG = 3, VERIFY NAME
? FIND GRADE = 7, STEP = 1, LET PAY = 10598, REPEAT 100
? FIND PART = X3513, VERIFY NAME, LET COST = 1.73,
  SOURCE = A.B. COOK
? FIND A/B = 0.1263, VERIFY FORMULA, LET A = 0.981
? FIND MOL = HCN, LET CSUBP = 1.5983, VERIFY ENTROPY, ID#
```

Our experience with other people's files has led us to choose a partial string search rather than an exact match in this module in order to avoid the trouble caused by inconsistent uses of blanks, parentheses, and punctuation in name and title fields. Thus, if this module were asked to FIND NAME = SMITH, the names SMITHSON, GOLDSMITH, etc., would also be accepted. This situation can be avoided easily in this module by specifying one or two additional items for more unique location, such as date of birth, employee number, etc., as indicated above.

When updates are more systematic, the instructions can be streamlined as follows:

```
? FIND GRADE = 12, STEP = 1, LET PAY = 12174, REPEAT 500
?     12,     2,     12580
?     12,     3,     12986
?     etc.
?     12,    10,     15828
?     13,     1,     14409
?     etc.
?     END
```

If the above pattern is broken in any one regard, that instruction must be written out in detail. The purpose of the REPEAT instruction is to ensure that changes are made in all records meeting the FIND specifications. Thus, the number following the word REPEAT must be as large as or larger than the number of anticipated occurrences of the specified records. An asterisk used instead of a number is equivalent to an infinite number of changes.

It is possible to streamline the above input still further by a rearrangement of the input as follows:

```
? FIND STEP = 1, LET PAY = 12174, FIND GRADE = 12, REPEAT 500
? 2, 12580
? 3, 12986
? etc.
? 10, 15828
? 1, 14409, 13
? 2, 14909
? 3, 15400
? etc.
```

This module has a provision for entering new data items to agree with the original conditions of the file as to left or right adjusted or centered values. It does not, however, prevent one from replacing a number by a string of characters or vice versa, nor do we think it should. If the new data item is longer than the width of the field allotted to it, the data is entered truncated and a diagnostic is printed.

If the global STORE switch has been set as described in section 1.6.2 the updated records are copied on a scratch file until the update is completed or until 50 records have been updated. In the latter case, the user is asked if he wished to continue to build a separate file of updated records. If the response is NO, the module writes no more records. If the answer is YES, at the end of the update the user has the option of fetching in the file of updated records and then displaying them.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >ftactmtg
* FILE FTACTMTG CONTAINS 17 DATA ITEMS FOR 1235 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >updateseq
*
*   *TYPE IDENTIFICATION AND UPDATE INSTRUCTIONS --->
* 1 ? >find rec = 5032,let nbspart = t
* 2 ? >find rec = 5033,let nbspart = t
* 3 ? >find rec = 5038,let nbspart = t
* 4 ? >find rec = 5039,let nbspart = t
* 5 ? >find rec = 5040,let nbspart = t
* 6 ? >find rec = 5041,let nbspart = t
* 7 ? >find rec = 5042,let nbspart = t
* 8 ? >find rec = 5047,let nbspart = t
* 9 ? >find rec = 5076,let nbspart = t
* 10 ? >find rec = 5088,let nbspart = t
* 11 ? >end
* UF31791030
* END OF FILE REACHED-NOT ALL DESIRED UPDATES MADE
* 2 UPDATES MADE. RECORD FOR UPDATE 3 NOT FOUND
*   *SHALL WE SKIP NUMBER 3 AND READ THROUGH
* THE FILE AGAIN, BEGINNING WITH 4 --->? >yes
* END OF FILE REACHED-NOT ALL DESIRED UPDATES MADE
* 4 UPDATES MADE. RECORD FOR UPDATE 6 NOT FOUND
*   *SHALL WE SKIP NUMBER 6 AND READ THROUGH
* THE FILE AGAIN, BEGINNING WITH 7 --->? >yes
* END OF FILE REACHED-NOT ALL DESIRED UPDATES MADE
* 5 UPDATES MADE. RECORD FOR UPDATE 8 NOT FOUND
*   *SHALL WE SKIP NUMBER 8 AND READ THROUGH
* THE FILE AGAIN, BEGINNING WITH 9 --->? >yes
* ALL 7 UPDATES MADE AFTER 38 RECORDS READ
*
* * * * *

```

Figure 3.4a. Here we see the dialogue with the UPDATSEQ module while attempting to update 10 records when three of those, indicated at [A], are missing from the file. At [B] we see how the file of updated records is named automatically. The conversation at points marked [C] shows how the module is instructed to skip over the missing records. The next figure shows that the records reported missing are indeed not in the file.

```

* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >display
* *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >rec,nbspart
*
* REC#   REC       N
* ---   ---       -
* 1     5001      0
* 2     5002      A
* 3     5003      A
* 4     5004      A
* 5     5005      A
* *MORE --->? >skip to 20
* *MORE --->? >yes
* 20    5032      T
* 21    5033      T
* 22    5034      A
* 23    5035      A
* 24    5036      A
* 25    5037      A
* 26    5039      T
* 27    5040      T
* 28    5042      T
* 29    5043      A
* *MORE --->? >yes
* 30    5044      A
* 31    5059      0
* 32    5060      A
* 33    5073      0
* 34    5074      A
* 35    5075      A
* 36    5076      T
* 37    5087      0
* 38    5088      T
* 39    5089      A
* *MORE --->? >stop
* PROGRAM STOPPED.
* * * * *

```



Figure 3.4b. Here we display the record numbers and the NBSPART vector to show that corrections had indeed been made as desired and that records 5038, 5041, and 5047 are indeed missing from this file. Note how the label NBSPART had been truncated to N since only one character was assigned to that data vector.

When there are a large number of updates to be made to a file, especially when they are systematic, like posting current salaries after an across-the-board pay raise, there is a more efficient way of updating using the module UPDATESEQ. That module expects the update instructions to be ordered to agree with the order that the offending records appear in the file. This permits all the updates to be carried out in a single pass through the main file as well as the update instructions. Items out of order in the update file are skipped. They are, however, reported at the end of the operation and the user is given an opportunity to make another pass through the file.

Even when the updates are not as systematic as those shown above, it is reasonable to expect that the data base manager will know enough about the location of the offending records to arrange the update accordingly.

If the file being updated contains checksums, the UPDATE module recomputes the two checksums, and replaces the existing information with the new numbers, as well as the current date, time, and run-id. The information provides an important audit trail — telling when and by whom the record was updated.

Figures 3.4a and 3.4b show how this module handles an update operation when some of the records referenced are not in the file.

\* \* N O T E S \* \*

### 3.5 Operating Omnidata in the Batch Mode and Remote Batch Mode

The discussion and illustrations of the workings of the Omnidata system have, thus far, involved the use of the Omnidata system from a terminal on-line in the so-called demand mode. When the computer is not burdened, the response is fast and often dramatic. This is especially so when the results appear on a scope as well as on a hard copy device (a teleprinter). It should now be obvious to the reader that we have gone to considerable length to make the system highly conversational—perhaps even uniquely so—and truly interactive. The conversational facility is especially valuable to the non-computer-oriented user, while the interactive features are of value to both the novice and the expert. The importance of the interactive characteristics of this system will become more apparent when applied to the statistical and graphical analysis and the manipulative facilities in the various utility modules.

It is nevertheless an inescapable fact that on-line use of any computer is more expensive than the batch mode of operation; even in the latter mode deferred batch (after normal business hours or overnight service) is cheaper yet.

Fortunately it is possible to operate the Omnidata system in any of the following modes:

- a) in demand mode with all input from the terminal — including, even, on-line file building;
- b) in demand mode from lengthy instruction sequences (keyboard inputs) that are used often or periodically, stored on a file (see 5.18, the PLAN module);
- c) in batch mode with inputs on punched cards;
- d) in the remote batch mode where the inputs are from the terminal but execution is deferred.

The Omnidata system has taken advantage of those features of the EXEC 8 system and of the XBASIC compiler that permit the above listed flexibility of operation.

Aside from cost, the first two modes of operation are most convenient and require no information concerning the operation of the EXEC 8 system beyond the normal log-on procedure. To run Omnidata successfully in the batch mode the user must know how to use certain EXEC 8 commands and must be very familiar with the step-by-step operation of the Omnidata modules he will use in the run.

In figure 3.5a we see a portion of the preamble to a batch mode run as it came off the line printer. Batch mode Omnidata runs are characterized by the fact that the users' responses do not appear on the same line with the question.

This run was initiated via 11 punched cards which contained the following:

```
@RUN,M/RBREENM,30305-BREENB,BASICEXT,20,200
@XBASIC,O
OLD:OMNIDATA
RUN
XXXXXX(account #)
XX (password)
MONO-INORG(a file)
STACK (a module name)
MONO-ORG(a second file)
TMONO (a name for the new file)
SAVE (a module name)
etc.
```

The first thing of interest here is that we called @XBASIC,O instead of @XBASIC. The O option causes the above user inputs to be printed out on the line following the Omnidata request. Without the O option these user inputs would not be seen at all.

The fourth mode of operation mentioned above is remote batch, which entails the user storing the commands for the Omnidata run in an element of a file while on line, starting that run from his terminal via an EXEC 8 command, and then receiving the output at the computer site, as in the ordinary batch mode just described. This is useful if, for example, while running in demand mode the user thinks of an Omnidata run he'd like which will produce voluminous output. By starting a remote batch run the results will be produced on a high speed printer, and the user will not have to sit at his terminal waiting for the lengthy printout. Nor will it cost as much.

The same results as appear in figure 3.5a would be achieved by the following sequence typed in on-line as a remote batch run:

```
@ELT,D1 BATCH.STACK (assuming 'BATCH' is the name of a
previously assigned file)
@XBASIC,O
OLD:OMNIDATA
RUN
XXXXXX
XX
MONO-INORG
STACK
```

```

MONO-ORG
TMONO
SAVE
etc.
@END
@START BATCH.STACK

```

An advantage of such a remote batch run is that the file element is permanently stored. Minor changes can be made (using the EXEC 8 @ED processor) after which the run can be restarted to produce different results.

```

* * * * *
* @run,m/r breenm,30305-breenb,basicext,20,200 *
* @xbasic,0 *
* XBASIC R5.1 16:21:05 15 APR 76 *
* old:omnidata *
* READY *
* run *
* OMNIDATA 16:21:06 15 APR 76 *
* *PLEASE ENTER ACCOUNT NUMBER --->? *
* >XXXXXXXXXX *
* *TYPE PASSWORD --->? *
* >XXXX *
* GOOD MORNING, WELCOME TO OMNIDATA *
* * *NOTE-OMNIDATA KEEPS A RECORD OF WHO USED WHICH *
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY * *
* *WHICH DATA BASE DO YOU WANT --->? *
* >mono-inorg *
* FILE MONO-INORG CONTAINS 18 DATA ITEMS FOR 2278 RECORDS. *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* --->? >stack *
* *TYPE FILES TO BE STACKED. *
* (NOTE:ONE IN CORE ASSUMED FIRST.) --->? *
* >mono-org *
* *TYPE NAME FOR NEW COMBINED FILE --->? *
* >tmono *
* *
* CPU SEC IN STACK = 7.6116 *
* CPU SEC = 14.6702 TIM = 16:25:29 *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* --->? >save *
* *
* * * * *

```

Figure 3.5a. This is a record of an Omnidata run made in the batch mode from a deck of cards shown on the facing page. Note that the user must anticipate all of the questions that the system and the various modules ask.

• • • N O T E S • • •

#### 4. An In-Depth Application of Omnidata to the NBS Crystal Data File

In what follows we shall see how the Omnidata system has been applied in a real working environment to a data file containing 18 data items for each of 24,000 crystal compounds comprising the NBS Crystal Data Determinative Tables [14]. That publication consists of two volumes: one for organic compounds and one for inorganic compounds. This logical division is carried over to the magnetic tape version and to the on-line data file on which we shall operate.

Each of the above mentioned volumes is further subdivided into six chapters in which the compounds are grouped into the standard crystallographic systems: monoclinic, cubic, tetragonal, hexagonal, orthorhombic, and anorthic. The on-line data file has been divided similarly so that the entire crystal data file is made up of 12 separate files suitably named and catalogued and identically formatted. They are called:

ANOR-ORG	MONO-ORG	ORTHO-ORG	TETRA-ORG
HEX-ORG	CUBIC-ORG	ANOR-INORG	MONO-INORG
ORTHO-INORG	TETRA-INORG	HEX-INORG	CUBIC-INORG

Dividing a large file into smaller subfiles makes for more efficient computer operations when searches can be restricted logically to certain portions of the file. It is our view that many existing monolithic files could be broken down logically into smaller mutually exclusive segments. The Omnidata system provides a number of modules to assist in file division as well as a tandem mode of operation when it is necessary to search through all of the subfiles or when an analysis is required for the file as a whole.

There are a number of other features in Omnidata to speed up operations when no logical way of breaking up a file exists but the file can be kept in some fixed logical order. These features will be discussed later.

Computerized data systems require the user to bring to them more or less knowledge about the stored data. The more user-oriented the system is the less does the user need to know about the data file to use it effectively. The Omnidata system requires a minimum of such information. The user of Omnidata need not be concerned at all with any of the following:

- a) programming languages (high level or low);
- b) the way the information is stored internally in the computer;
- c) the order in which the data items are stored for each of the records in the file; or
- d) how the logical records are related to the physical record or to the disc sectors.

```

* * * * *
* >old:omnidata*
* READY
* >run
*   OMNIDATA      15:32:54      17 MAY 77
*   *PLEASE ENTER ACCOUNT NUMBER --->? >XXXX
*   *TYPE PASSWORD --->? >XXX
*   *WHICH DATA BASE DO YOU WANT --->? >anor-inorg
* GOOD AFTERNOON, WELCOME TO OMNIDATA
* ***NOTE - OMNIDATA KEEP A RECORD OF WHO USED WHICH***
*   MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*
* FILE ANOR-INORG CONTAINS 18 DATA ITEMS FOR 358 RECORDS.
*
*   *TYPE A MOULE NAME AND/OR INSTRUCTIONS
*   --->? >labels
*
* THE FILE ANOR-INORG CONTAINS DATA LABELLED AS FOLLOWS:
*
* 4   A           7   ALPHA      5   B           8   BETA          6   C
* 11  DM          12  DX          14  FORMULA      9   GAMMA         2   I OR O
* 15  IDNUM       16  M OR N     18  NAME         3   R 1          17  R2
* 13  SG          1   SYSTEM     10  Z
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   --->? >modules
*
* THE FOLLOWING MODULES ARE AVAILABLE:
*
*  ABRIDGE      AGGREGATE      ANALYSIS      ANNEX
*  ARRAY        ATTACH        BLANKS        BROWSE
*  CHECKSUM     COMPUTE        CONCAT        CROSSTAB
*  DESCRIBE     DICTIONARY     DISPLAY       DISTRIBUTUTE
*  ENCODE       EXTRACT        FETCH         FIT
*  GRAPH        KWOC          MOVE          PLAN
*  PLOT         RANDOM        REGRESS       RENAME
*  REPORT       SAVE          SCREEN        SEARCH
*  SEGMENT      SEQUENCE       SORT          STACK
*  STATIS       STATPLOTS     SUMMARY       SURVEY
*  TALLY        TRIM          UPDATE        UPDATESEQ
*  USERS

```

Figure 4 a. See the text for a discussion of the marked items in this figure.

The file builder or manager, on the other hand, does need to know the information in items c and d above in order to define the file into the Omnidata format. How this is done is explained in section 2.1.

Returning to the file user again, the reader is reminded that in order to make use of the Omnidata system a user must know the name of the file he wishes to use and must, furthermore, be accredited as a user of that file. In figure 4a we see an introductory dialogue between Omnidata and the user of one of the crystal data files (ANOR-INORG).

At [A] we see how the user starts the cycle by asking for the main program Omnidata under the SBASIC compiler. The compiler responds at [B] with the date and time the run was started. The subsequent responses are from the main Omnidata program which requests the user's account number at [C], a password at [D], and a file name at [E]. These responses are checked against the designated file to see if the user is accredited to that file. If so, he is allowed to proceed. If not, a suitable diagnostic is printed and what happens next depends upon which of the responses cannot be reconciled with the accreditation data in the requested file. These options, which are discussed in section 1.6.3, allow for efficient recovery from inadvertant typing errors, while precluding access to unauthorized persons.

Having passed the security checks built into the data file, the user is greeted at [F] and informed at [G] that Omnidata keeps detailed records of file and system users; he is further informed at [H] that the requested file contains 18 data items for each crystal (record) and that there are data for 358 crystals in that file.

At this stage the Omnidata program is ready to respond to a request for a particular module or to carry out one or more of the global instructions discussed earlier. The response at [I] provides the user with the labels associated with the crystal data file. These are the names of the data items to which the system will respond for searching or other operations. The numbers associated with the labels can be used interchangeably with the names. The labels are numbered in the order in which they were defined when the file was set up. At [J] we see again the request for a module name or instructions and the response MODULES, which produces the list of active Omnidata modules of concern to file users.

As illustrated thus far the system has supplied the user with general information about the system and the general content of the ANOR-INORG data file. To be able to search that file and others in the crystal data file system, the user needs to know how the detailed information is entered in the file. The user will need to know:

```

*****
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? >display
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*   -->? >all
*
*   RECORD NUMBER 1
*   SYSTEM -- A
*   I OR 0 -- I
*   R1 -- 0.0221
*   A -- 5.3
*   B -- 240.0
*   C -- 5.3
*   ALPHA -- 90.00000
*   BETA -- 120.00000
*   GAMMA -- 90.00000
*   Z -- 24.
*   DM --
*   DX --
*   SG -- P-1.
*   FORMULA -- KR3(O H, F)2( AL, SI)4 O10
*   IDNUM -- 000001
*   M OR N -- MI
*   R2 -- 0.0221
*   NAME -- POTASSIUM HYDROXIDE-FLUORIDE ALUMINATE-SILICATE (1+2+4+10)
*
*   *MORE --->? >no
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*   --->? >system,i or o,dm,sg,m or n
*
*   REC #   S   I   DM           SG           M
*   -----
*   2       A   I           P-1.         MI
*   3       A   I           P*.         NM
*   4       A   I           P-1.         MI
*   5       A   I   2.147         MI
*   6       A   I   2.92          MI
*
*   *MORE --->? >5
*   7       A   I   5.329         P-1.         MI
*   8       A   I   6.01          P-1.         MI
*   9       A   I   2.702         P-1.         MI
*   10      A   I   2.699         P1.          MI
*   11      A   I   2.71          P1.          MI
*
*   *MORE --->? >yes
*****

```

Figure 4 b. See the text for a discussion of the marked items in this figure.

- a) how are space groups indicated?
- b) how are the chemical formulas written?
- c) how are angles entered?
- d) how are chemical names written?
- e) etc.

Omnidata provides a considerable variety of ways to obtain that information via modules which have their major utility quite apart from providing answers to the above questions. The first of these is the DISPLAY module, whose operation is illustrated briefly in figure 4b, et seq., and more fully in section 3.2.

The DISPLAY module allows one to see easily all of the data items and their labels for the first record in the file and for as many succeeding records as desired. At point [A] in figure 4b, this module asks for either the word ALL or a list of labels (names of data items) to be displayed. Since the response at [B] is ALL, we are shown the 18 pieces of information stored in the first record of this file. From this printout we see: at [C] that the determinative ratio R1 is entered to four decimals; at [D] that the angles (alpha, beta, and gamma) are in degrees to five decimals; at [E] that entries for the densities DM and DX are blank in this record; at [F] how the chemical formula is entered in the record; and at [G] that the crystal is a mineral (MI).

Although this form of the DISPLAY module provides useful information to one unfamiliar with the detailed way in which the data is recorded in the file, it does not answer all questions on this subject. While we see at [G] that minerals are tagged as MI, we do not yet know how non-minerals are indicated, nor can we be sure that the formula given in this record is representative of the entire file.

To get a look at a larger sample of the data in this file, we respond at [H] with the word NO. This changes the operation of this module from the ALL mode to the alternate mode in which we can ask for specific data items to be displayed. At [I] we request a display of the data items SYSTEM, I OR O, DM, SG, M OR N, and the result is a table displaying the above data items for the next five records in the file. On careful examination we see at [J] the entry NM which we can safely assume to be the way in which nonminerals are tagged in the data vector labeled M OR N. The reason why only the letter M is printed as the heading of this column is explained in section 3.2. At [K] we see missing data items in the DM and SG columns. At [L] we have an opportunity to see some more of the data in this file and note from the next five records some representative space group (SG) entries and the variation in number of decimal in the vector DM. The response at [L] yields five more records.

```

* * * * *
* 12      A      I      3.0                P-1.          NM
* 13      A      I                P1.           MI
* 14      A      I      3.46              P*            MI
* 15      A      I      2.154             MI
* 16      A      I      2.181             P-1.          MI
* 17      A      I                        NM
* 18      A      I      2.18              P1.           MI
* 19      A      I      4.15              P-1.          MI
* 20      A      I                        MI
* 21      A      I      2.33              MI
*
* *MORE --->? >no (M)
* *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >formula (N)
*
* REC      FORMULA
* -----
* 22
* 23 (O) (FE, MN, CA, MG) SI O3
* 24 (MN, FE) SI O3
*
* 25      BA2 MN2 (TI, FE) O(S12 O7) (P, S) O4 O H
* 26      (BA, SR, NA) 2 (MN, FE, CA, MG) 2 (TI, FE, AL) O(O4)(O H)
*
* *MORE --->? >back to 1 (P)
* *MORE --->? >yes
* 1      K R3(O H, F) 2 (AL, SI)4 O10
*
* 2      K R3(O H, F) 2 (AL, SI)4 O10
* 3      YB6 TI O11
* 4      PB3 AS4 S9
* 5      FE2(S O4)3+10 H2 O
* 6      CA SI O3
* 7      PB3 AS4 S9
* 8      PB5 SN3 SB2 S14
* 9      MG5(FE, CR, AL) (SI, AL)4 O18 H8
* 10     (MG, FE, AL) 6 (O H) 8 (SI, AL)4 O10
*
* *MORE --->? >skip to 24 (Q)
* *MORE --->? >yes
* 24     (MN, FE) SI O3
* 25     BA2 MN2(TI, FE) O(S12 O7) (P, S) O4 O H
* * * * *

```

Figure 4 b (concluded). Note how the DISPLAY module allows one to skip back and forth in a file. This feature is useful to locate on a particular record as shown in figure 5.10a.

Finally, a response of NO at [M] allows us to display some other data vectors, and we elect at [N] to see how the formulas are entered. We are quickly reminded at [O] that the DISPLAY module works its way down the file. As we wish to see the formulas for the earlier records in the file, we instruct the module at [P] to back up (BACK TO 1) and get as many records listed as we wish.

Another way to get a look at how data are entered in the file is via the BROWSE module. In figure 4c we see the result of browsing on the space group vector SG. This module displays each unique entry once as it finds it in the space group field while it searches through the file. Each time a new item is found the record number in which it occurred is printed. Thus, we can see at [B] that the first two records belong to space group P21, that there is a blank in the SG field in record three, etc. This operation proceeds until 10 unique space groups are reported, at which time we have an opportunity to continue or not, as we do at point [C]. At [D] we forego a report of how many records contained each of the listed space groups. This feature is illustrated and discussed further in section 5.05.

```

*****
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*  --->? >browse (A)
*
*      WHICH LABEL DO YOU WANT TO BROWSE ON --->? >sg
*  LINE      REC NO.  SG
*
*  1.         1        P21. (B)
*  2.         3        *BLANK*
*  3.         4        PC.
*  4.         5        P21/N.
*  5.         9        C*/C.
*  6.        12        P21/C.
*  7.        14        P21/M.
*  8.        22        P2/M.
*  9.        26        P21/*.
* 10.        30        P21/A.
* 11.        31        AA.
* 12.        36        I*/A.
* 13.        41        CC.
* 14.        42        A2/M.
* 15.        45        A*/A.
*
*  *MORE --->? >5 (C)
*
*  *DO YOU WANT A TALLY PRINTED OUT --->? >no (D)
*  *WHICH LABEL DO YOU WANT TO BROWSE ON --->? >stop (E)
*****

```

Figure 4c. In this output from the BROWSE module the record numbers show the first record in which the particular space group appears.

From the question at [E], we see that we now have an opportunity to BROWSE on a different data vector. We STOP the run, instead, in order to choose a different subfile so as to illustrate another feature of the BROWSE module. In figure 4 d we see that each time the BROWSE module has read 100 records without finding a new entry to post, it tells how far through the file it has progressed and asks if it should continue. Since the response at [B] is negative, we are afforded an opportunity at [C] to see how many of each of the entries were found in the records read up to that point. In this table the numbers 1 to 15 refer to line numbers in the results above and are to be interpreted as follows: 2 records contained a Z value of 24; 35 records were blank; 41 records contained a value of 1 for Z; etc.

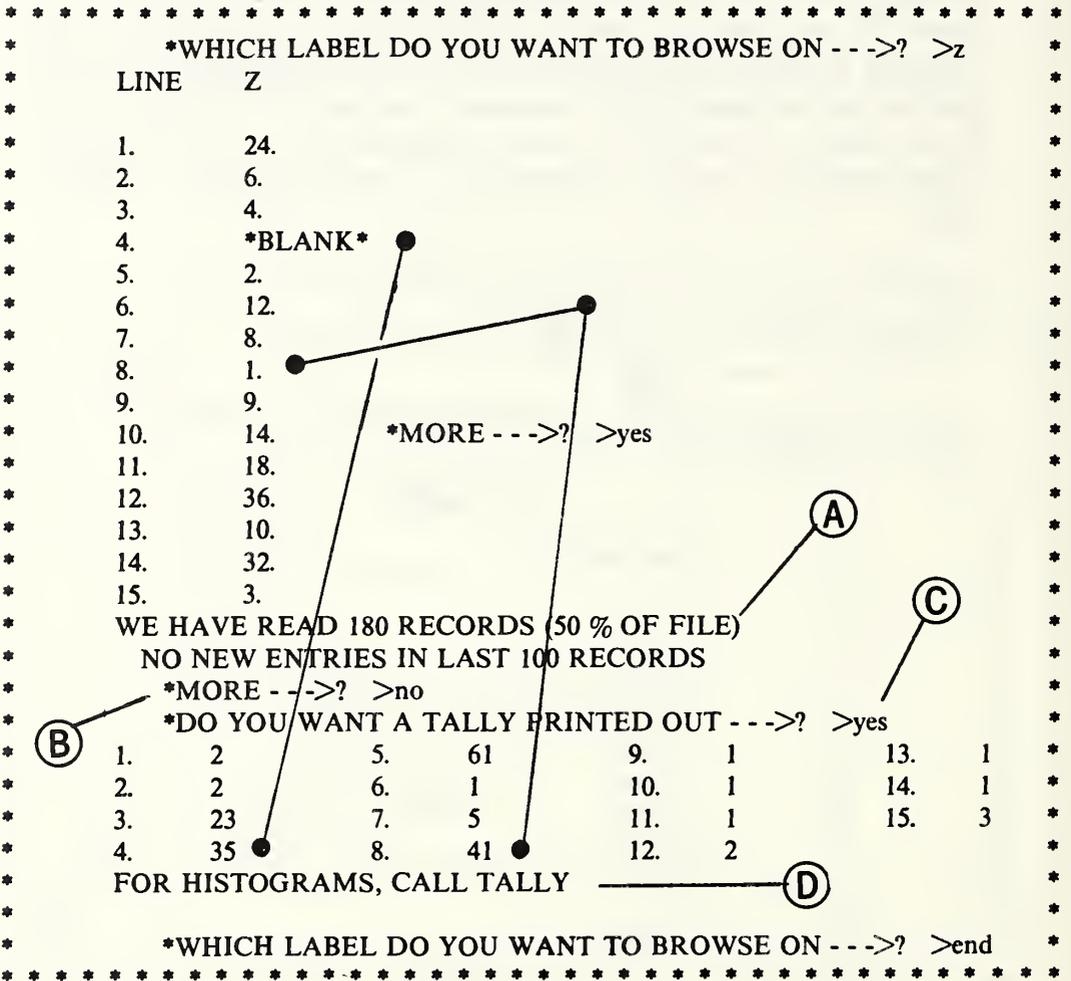


Figure 4 d. Here we see how the BROWSE module keeps and reports a tally of how many of each of the values there are in the designated data vector. While the first blank occurred in record 4, there are actually 35 blank entries in the space group (SG) column in the 180 records read thus far. See the text for additional comments on this figure.

The reminder at [D] tells us of still another way to look at the totality of unique entries in a data vector, namely, via the module TALLY illustrated in figure 4e. An explanation of the dialogue shown at [A] is deferred until section 5.34. The TALLY module is of great utility to the data file manager because it provides an easy way of checking on missing data items, on typographic errors, or inconsistencies in the data entries. Prior to the utilization of the AGGREGATE module on the crystal data file (discussed later in this section), a run through the TALLY module showed that 4 of the 184 different entries in the SG vector in the orthorhombic system were to nonexistent space groups. These errors were subsequently traced back to typographic errors in the original sources of the data.

Early in the development of this module, attempts at providing a TALLY on job titles revealed such a host of variations (legitimate and otherwise) as to make such a listing almost useless. This state of affairs did, however, motivate us to build a number of interesting and highly useful features into this module. In section 5.34 we show how tallying on the first word (or character) or the first n words (or n characters) from the front of a data field (or from the back) can yield interesting results when applied to hierarchically structured character strings within individual entries in a data field.

Now that we know how space groups and chemical formulas are handled in the crystal data file we can proceed to a discussion of how the SEARCH module is used. In figure 4g we see a record of the dialogue between Omnidata and a user who wishes to see certain data on all inorganic compounds in the anorthic system containing sodium and either nickel or copper or iron. After calling the proper disk file at [A] and instructing the SEARCH module to operate in the SELECT mode at [B], the search criteria are supplied at [C]. Upon completion of the search we learn at [D] that 10 records satisfy the criteria. At [E] we select the PRINT option on the resulting *current* file and ask to see the five data items indicated at [F]. As we wish to use the REPORT module for the rest of the data items, we exit at [G] with the current file.

While the DISPLAY module is intended for a quick look at data, the REPORT module is intended for preparing finished reports. Thus, ample provision has been made in this module for fancy formatting. Among these are provisions for page headings and footings, column headings for tabular arrangement, and selective labeling of data. This module works in two major modes — labeled and unlabeled, but in each mode it is possible to exempt certain data items. In figure 4h we have an example of how a small report is designed using the interactive feature of this module to position the column heading precisely where desired.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT -->? >anorthic
*
*   GOOD AFTERNOON, WELCOME TO OMNIDATA
*   ***NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH***
*   MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*
*   FILE ANORTHIC CONTAINS 18 DATA ITEMS FOR 1026 RECORDS.
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? >tally
*
*   *TYPE LABELS (UP TO 3) TO BE TALLIED; FOLLOWED BY END
*   -->? >sg,sg,end
*   *TYPE ANY SEPARATORS YOU WISH RECOGNIZED, OR N(ONE)-->
*   ? >none
*
*   *DO YOU WANT THE TALLY OF SG SORTED
*   (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*   OR (DESCEND)ING ORDER -->? >alpha
*   *SHALL WE NORMALIZE -->? >yes
*   TALLY OF SG
*
*   SG          CUM %      % CUM FREQ  FREQ
*   -----
*   P-1.        31.2       31.2    320    320 XXXXXXXXXXXX
*   P*.         88.2       57     905    585 XXXXXXXXXXXXXXXXXXXX
*   P1.         92.80      4.6    952    47 X
*   ? >         100       7.2   1026   74 XX
*
*   *DO YOU WANT THE TALLY OF SG SORTED
*   (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*   OR (DESCEND)ING ORDER -->? >descend
*   *SHALL WE NORMALIZE -->? >yes
*   TALLY OF SG
*
*   SG          CUM %      % CUM FREQ  FREQ
*   -----
*   P-1.        57        57     585    585 XXXXXXXXXXXXXXXXXXXX
*           88.2       31.2    905    320 XXXXXXXXXXXX
*   P1.         95.4       7.2    979    74 XX
*   P*.         100       4.6   1026   47 X

```

Figure 4 e. Here we TALLY the space group field twice to illustrate two arrangements — alphabetically at [B] and according to frequency at [D]. The input request at [C] provides a pause to allow for positioning of the paper after which a carriage return will signal the module to produce the next tally. See section 5.34 for an explanation of the request at [A] and to learn how the histograms look when not normalized.

```

*****
*DO YOU WANT THE TALLY OF SG SORTED
(ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
OR (DESCEND)ING ORDER -- ->? >descend
*SHALL WE NORMALIZE -- -> ? >yes
*HOW MANY OF THE 47 ENTRIES DO YOU WANT PRINTED--->? >all

```

SG	CUM %	% CUM	FREQ	FREQ	
	26.2	26.2	1238	1238	XXXXXXXXXXXXXXXXXXXXXXX
FM3M.	(A) 47.2	21	2229	991	XXXXXXXXXXXXXXXXXXXX
FD3M.	60.9	13.8	2880	651	XXXXXXXXXXXX
PM3M.	73.10	12.1	3454	574	XXXXXXXXXXXX *
F-43M.	76.30	3.2	3604	150	XX *
PA3.	79.30	3.1	3750	146	XX *
IA3D.	82.10	2.8	3882	132	XX *
I-43D.	83.9	1.7	3964	82	X *
IM3M.	85.60	1.7	4045	81	X *
P213.	87.30	1.7	4125	80	X *
PM3N.	88.80	1.5	4196	71	X *
IA3.	90	1.2	4253	57	X *
1-43M.	91.10	1.1	4305	52	X *
FM3C.	92	.9	4349	44	X *
PN3M.	92.9	.8	4389	40	X *
P-43M.	93.7	.8	4429	40	X *
F*3*.	94.60	.8	4469	40	X *
FM3.	95.2	.6	4497	28	*
PN3.	95.7	.6	4523	26	*
IM3.	96.2	.5	4548	25	*
F***.	96.7	.5	4571	23	*
F-43C.	97.2	.5	4594	23	*
P41,332.	97.7	.4	4615	21	*
P-43N.	97.9	.2	4626	11	*
I***	98.10	.2	4636	10	*
F23.	98.30	.2	4646	10	*
I213.	98.5	.2	4655	9	*
PN**.	98.7	.2	4663	8	*
P4232.	98.80	.2	4671	8	*
P*3*.	99	.1	4678	7	*
I23.	99.10	.1	4685	7	*
I*3*.	99.2	.1	4690	5	*
I4132.	99.30	.1	4695	5	*
P23.	99.4	.1	4700	5	*
FD3.	99.60	.1	4705	5	*

Figure 4 f. Here we have results of a tally of space groups in a larger file. Note that when the number of unique items found in the designated data vector exceeds 10, the user is informed at [A] what that number is and asked how many are to be printed.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT -->? >anor-inorg (A)
*
*   GOOD AFTERNOON, WELCOME TO OMNIDATA
*   ***NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH***
*   MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*
*   FILE ANOR-INORG CONTAINS 15 DATA ITEMS FOR 358 RECORDS.
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? >search (B)
*
*   *TYPE S(ELECT) OR R(EJECT) -->? >select
*   *TYPE LABEL AND VALUE(S)
*   1 -->? >formula,na (C)
*   AND
*   2 -->? >formula,ni or cu or fe (C)
*   AND
*   3 -->? >end
*   10 HIT(S) WHEN: (D)
*   FORMULA IS NA
*   AND
*   FORMULA IS NI OR CU OR FE
*   *TYPE P(RINT),S(EARCH) OR E(XIT) -->? >print (E)
*   *TYPE C(URRENT),P(REVIOUS), OR O(RIGINAL) -->? >current
*   *TYPE LABELS OF ITEMS TO BE PRINTED OR ALL
*   -->? >r1, a, b, c, dm (F)
*
*   R1          A          B          C          DM
*   0.4775      7.01      14.68     5.38      4.19
*   0.8225      8.20      9.97      6.97      2.74
*   0.8690      10.55     12.14     8.00      3.66
*   0.8885      10.12     11.39     7.27
*   0.9372      9.752     10.406    8.926     3.81
*   0.9558      9.72      10.17     9.56     3.378
*   0.9687      9.3       9.6       9.2       1.64
*   0.9756      12.82     13.14     5.42     3.498
*   0.9815      11.660    11.880    5.391     3.42
*   0.9841      11.75     11.94     5.35
*
*   *MORE -->? >no
*   *TYPE P(RINT),S(EARCH) OR E(XIT) -->? >exit (G)
*   *TYPE C(URRENT),P(REVIOUS), OR O(RIGINAL) -->? >current
* * * * *

```

Figure 4 g. Here we see the result of searching for crystals containing sodium (NA) plus either nickel (NI), copper (CU), or iron (FE) in the chemical formula. The PRINT option is not a separate module; it provides a quick way of looking at data while in the SEARCH module.

In this example, we request at [A] an unlabeled report with 20 lines 56 characters wide. Had we omitted the number of lines in the above instruction, the page depth would be formatted automatically to be proportional to a 6x9 inch print area. At [B] we indicate the data items to be printed and the title of the report, while at [C] we are content to use the assigned labels as column headings. At [D] we are shown how the column headings would appear over the tabulated data and are given an opportunity at [E] to make changes in the position or even the wording. At [F] we choose simply to shift the headings to center them over the columns, while at [G] we are shown again how the modified headings look. We accept this modification, and after answering two more questions in the negative at [H], the report is printed as instructed.

In figure 4i we see another application of the REPORT module to print a short table in which the formula field and the name field are presented for the first 19 records in the cubic-organic file. Here we show a feature of REPORT which allows us to specify at [A] that only the first 20 characters of the formula field and only the first 35 characters of the name field should be printed in the report. The use of the SHORT option allows us to print both the formula and name on the same line even though 80 characters are reserved for formulas and 90 characters are reserved for names in the actual file. We chose to display these 19 lines because we knew that the data in them was short enough to fit in the designated space. Had the names been longer, the instruction at [A] would have caused them to be truncated in the report.

In this report we supplied the column headings at B in such a way as to place them more nearly centered on their respective columns. Had we chosen to do so we might have used the words COMPOUND or COMPOSITION instead of the headings FORMULA or NAME. It should be noted that in both title lines and column headings we are not restricted to single line entries. Since each line is preceded by a line number, it is possible to change one's mind in the process simply by repeating a line number used previously and following it by the revised text. Other features of the REPORT module are discussed in section 5.23.

One of the advantages of having a large data file in computerized form is that it can be sorted in various ways on one or more key data elements. This is especially true in the case of the Crystal Data File where the data are arranged in 12 separate chapters and by chapter in increasing value of the first determinative ratio (R1). This logical organization is carried over to the computer files to reduce search time when information is desired from one or more specific crystal systems. When certain information is desired from all systems, it is not necessary to handle each file separately because the STACK module will put the desired files in a specified order so that Omnidata can treat them as a single file.

```

*****
*   *TYPE MODE- -LABELLED OR UNLABELLED --->?   unlabeled
*   *TYPE DIMENSIONS OF LINE AND PAGE --->?   56,20
*   *TYPE LINES FOR OUTPUT
*   LINE 1 ? r1, alpha, beta, gamma
*   LINE 2 ? end
*   *TYPE TITLE LINE OR N(ONE) --->
*   ? 1,crystals containing na and ni or cu or fe (part 2)
*   ? end
*   *ANY COL HEADS- -TYPE L(ABELS),N(ONE) OR 4 HEADINGS --->
*   ? r1,alpha,beta,gamma
*   ? end
*   R1          ALPHA          BETA          GAMMA
*   0.4775      93.50000      90.20000      95.30000
*   *OK- -TYPE YES OR NO --->?   no
*   *TYPE LINE NO., OLD HEAD, NEW HEAD --->
*   ? 1,r1 ';' r1
*   ? 1,alpha ';' alpha
*   ? 1,beta ';' beta
*   ? 1,gamma ';' gamma
*   ? end
*   R1          ALPHA          BETA          GAMMA
*   0.4775      93.50000      90.20000      95.30000
*   *OK- -TYPE YES OR NO --->?   yes
*   *TYPE FOOTNOTE LINES OR N(ONE) --->
*   ? none
*   *DO YOU WANT TO POSITION PAPER AFTER EACH PAGE --
*   TYPE YES OR NO --->?   no
*
*   CRYSTALS CONTAINING NA AND NI OR CU OR FE (PART 2)
*   -----
*   R1          ALPHA          BETA          GAMMA
*   =====
*   0.4775      93.50000      90.20000      95.30000
*   0.8225      98.96667      114.78333      105.03333
*   0.8690      105.06667      96.38333      107.35000
*   0.8885      91.35000      99.05625      111.08958
*   0.9372      96.86667      114.41667      64.78333
*   0.9558      109.30000      91.30000      71.00000
*****

```

Figure 4 h. In this application of the REPORT module we show how it allows the user to supply column headings and to edit them. Since the headings happen also to be the assigned labels, a response of LABELS at [C] would cause them to be centered automatically. See the text for a further discussion of this figure.

```

* * * * *
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? report
*
*   *TYPE MODE- -LABELLED OR UNLABELLED -->? unlabeled
*   *TYPE DIMENSIONS OF LINE AND PAGE --->? 56, 25
*   *TYPE LINES FOR OUTPUT
*   LINE 1 ? formula,short,20,name,short,35
*   LINE 2 ? end
*   *TYPE TITLE LINES OR N(ONE) -->
*   ? 1,first 19 records in the cubic-inorg file
*   ? end
*   *ANY COL HEADS- -TYPE L(ABELS),N(ONE) OR 2 HEADINGS- ->
*   ? '      formula,'      name'
*   ? end
*   FORMULA                      NAME
*
*   P                            PHOSPHORUS
*   *OK- -TYPE YES OR NO --->? yes
*   *TYPE FOOTNOTE LINES OR N(ONE) --->
*   ? none
*   *DO YOU WANT TO POSITION PAPER AFTER EACH PAGE --
*   TYPE YES OR NO --->? no
*
*   FIRST 19 RECORDS IN THE CUBIC-INORG FILE
*   -----
*
*   FORMULA                      NAME
*   =====
*   P                            PHOSPHORUS
*   BE                            BERYLLIUM
*   BE CO                          BERYLLIUM COBALT (1+1)
*   BE CU                          BERYLLIUM COPPER (1+1)
*   CO6 SI2.5 V1.5                COBALT SILICON VANADIUM (6+2.5+1.5)
*   BE PD                          BERYLLIUM PALLADIUM (1+1)
*   FE3 SI                         IRON SILICON (3+1)
*   AL NI                         ALUMINUM NICKEL (1+1)
*   CO( MN0.5 SI0.5)              COBALT MANGANESE-SILICON (1+1)
*   AL CO CU2                     ALUMINUM COBALT COPPER (1+1+2)
*   CO FE                         COBALT IRON (1+1)
*   ( FE, B)                      IRON-BORON
*   AL CO                         ALUMINUM COBALT (1+1)
*   MN3 SI                        MANGANESE SILICON (3+1)
*   ( FE, NI, CO)                 IRON-NICKEL-COBALT
*   FE                            IRON
* * * * *

```

Figure 4 i. Here we illustrate the use of the SHORT instruction when formatting a report. See the text for a further discussion of this figure.

While the STACK module is a convenience in searching, it is a necessity in sorting when one wishes, for example, to sort the data on space groups regardless of whether the crystals are organic or inorganic. In figure 4j we see at [B] how the STACK module is told to add the anorthic-organic file to the anorthic-inorganic file which was called up first at [A], while at [C] we provide a name for the composite file. At [D] we type the global command LENGTH to see how long the composite file is, to double check, and follow by calling the SORT module at [E]. The instructions at [F] are to sort alphabetically on space group and then on R1 within each space group. A discussion of the uses of two options indicated at [G] is to be found in section 5.28. At [H] we use the DISPLAY feature of the SEARCH module to see that the file has indeed been sorted as desired.

We have seen earlier how the TALLY module tells us a good deal about the distribution of unique data entries in a given data field. Another item of interest is the distribution of values in one data field among those in another. In particular, how are the values of Z in the file distributed among the space groups (SG)? Such information is provided by Omnidata via the CROSSTAB module illustrated in figure 4k. This module produces a two-way analysis, in this case, of Z values against space groups (SG). From this table we see that of the 78 records in which Z=1, 50 are in space groups P-1, 6 are space group P\*, 15 are in records where the space group is blank, and 7 are in space group P1. While from a crystallographic point of view such a cross tabulation is not very important, it can serve to identify typographic errors. Other applications of this module are illustrated in section 5.8.

At a certain stage in working with the crystal data file of all organics it became necessary to divide it into separate files for each of the crystallographic systems — cubic, hexagonal, orthorhombic, etc. That operation was performed by the DISTRIBUTE module as shown in figure 4l. We see there at [A] a request to monitor the operation each time 2000 records are processed and the result of that monitoring at [C]. When the operation is completed, we get a report at [D] giving the names by which the subfiles have been saved and how many records they contain.

The names are composed of three items: F for file; SYSTEM for the label of the data vector on which the file was split; and the letters A, M, O, T, H, and C which were the actual file entries for the crystal systems— anorthic, monoclinic, orthorhombic, tetragonal, hexagonal, and cubic, respectively.

```

*****
*WHICH DATA BASE DO YOU WANT --->? >anor-inorg (A)
FILE ANOR-INORG CONTAINS 18 DATA ITEMS FOR 358 RECORDS.

*TYPE A MODULE NAME AND/OR INSTRUCTIONS
--->? >stack
*TYPE FILES TO BE STACKED. (B)
(NOTE:ONE IN CORE ASSUMED FIRST.) --->
? >anor-org
*TYPE NAME FOR NEW COMBINED FILE --->? >anorthic (C)

CPU SEC IN STACK = 1.2686
CPU SEC = 1.5274 TIME = 16:01:06

*TYPE A MODULE NAME AND/OR INSTRUCTIONS
--->? >length (D)

PRESENTLY, YOUR FILE HAS 1026 RECORDS.
*TYPE A MODULE NAME AND/OR INSTRUCTIONS (E)
--->? >sort
*TYPE ALPHA(BETIC) OR VAL(UE) SORT --->? >alpha (F)
*TYPE SORT KEYS IN ORDER --->?. > sg,rl
*TYPE CHARACTERS TO BE IGNORED IN SORT KEY, OR NONE --->? >none
*TYPE LABELS TO BE TOTALLED AND AVERAGED --->? >none (G)
CURRENT FILE IS SORTED- -CALL SAVE TO SAVE.

CPU SEC IN SORT = 76.854
CPU SEC = 83.639 TIME = 16:06:33

*TYPE A MODULE NAME AND/OR INSTRUCTIONS
--->? >display
*TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
--->? > sg,rl

SG                R1
-----
P-1.              0.0221
P-1.              0.0883
P-1.              0.1168
P-1.              0.1270
P-1.              0.1481
P-1.              0.1552
P-1.              0.1626
P-1.              0.1671

```

Figure 4 j. Here we see an application of the STACK module followed by a SORT on the composite file. See the text for a discussion of the marked places in this figure.

Subsequent to the publication of the NBS Crystal Data Determinative Tables [14] it was decided to prepare a supplementary publication [15] to serve as an index to well-characterized and properly classified crystals reported in the published table. Here it was important to arrange the index first by crystal system (cubic, monoclinic, tetragonal, etc.) and then by space group. This would have been a trivial problem for Omnidata, not worth mentioning, were it not for the following complications:

- a) records in which space group entries were blank were to be deleted from the index;
- b) records in which the space groups were uncertain (those in which an asterisk appeared) were also to be deleted; and,
- c) the remaining space groups were to be reclassified into broader numeric classes.

It is this last requirement that affords us the opportunity to illustrate the use of the AGGREGATE module. Before doing so it seems worthwhile to review briefly the steps in solving this production problem for the monoclinic system via Omnidata.

Step 1. Here we combine the two monoclinic files (MONO-ORG and MONO-INORG) by using the STACK module.

Step 2. Next we call the SEARCH module and reject records from the file in which the space group is either blank or contains an asterisk (\*).

Step 3. The AGGREGATE module is used next to achieve reclassifications of the space groups.

Step 4. The resulting file is sorted on the new space group designations.

Step 5. The file is then printed, listing: the space group, the first determinative ratio R1, and the chemical formula.

The operation of the AGGREGATE module is shown in figure 4.1 where at [A] we instructed the main Omnidata to set the monitor switch to 500 so that the module will report each time it has read 500 records (as it indeed does at [E]). At [B] we designated by SG that it is the space group data entries which are to be aggregated, while at [C] we supply the name of the new data vector which the module will generate called SGNUM. The detailed instructions are supplied at [D] In the first 2 lines, we see how the module is instructed to enter the number 14 in the new data vector (SGNUM) for each record in which the SG is either P21/A., P21/A, P21/N., P21/N, P21/C., or P21/C. The spaces which follow these space group designations in the input stream are to allow for the blanks that follow these designations in the 10-character field reserved for the SG vector in each record. Other interesting features of the AGGREGATE module are discussed in section 5.2.

```

*****
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS      *
*  -->? >crosstab                                     *
*
*      *TYPE LABELS TO BE CROSSTABBED -->? >z,sg    *
* CAUTION, SG IS NO A NUMBER IN THIS FILE           *
*      *RESPOND WITH AN OPTION NUMBER.              *
*          1      ACTUAL VALUE                       *
*          2      % OF TOTAL IN FILE                 *
*          3      % OF TOTAL IN Z                   *
*          4      % OF TOTAL IN SG                   *
*          0      TO STOP                             *
*  -->? >1                                           *
*      *ANY HEADING -->? >distribution of z values among space groups *
*      *ANY FOOTNOTE-->? > NO                       *
*
*          DISTRIBUTION OF Z VALUES AMONG SPACE GROUPS
*
*          **** SG ****
*
*  Z          P-1.      P*.          P1.          TOTAL
*  -----
*  1          50         6           15          78
*  2          112        4           30          151
*  3          2          0           3           5
*  4          25         2           17          44
*  6          4          0           3           8
*  8          7          1           3           12
*  9          0          0           1           1
*  10         0          0           1           1
*  12         0          0           1           1
*  14         1          0           0           1
*  16         0          0           1           1
*  18         0          0           1           1
*  24         1          0           1           2
*  32         1          0           0           1
*  36         0          0           2           2
*  -----
*  TOTALS     203       13          78          309
*
*          THIS TABLE SHOWS ACTUAL VALUES
*****

```

Figure 4 k. In this application of the DISTRIBUTE module we have chosen to tabulate the actual values for the distribution of space groups among the Z values. Note that 78 records contain blanks in the space group field. If blanks had been encountered in the Z field they would have shown up on the first line before Z=1.

```

* * * * *
* FILE ALLORGANIC CONTAINS 18 DATA ITEMS FOR 7496 RECORDS.
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? >monitor,2000 (A)
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? >distribute (B)
* *DISTRIBUTE ON WHICH LABEL -->? >system
*
* 2000 ENTRIES DISTRIBUTED TO 2 FILES 4.6904 105818
* 4000 ENTRIES DISTRIBUTED TO 2 FILES 9.051 110009
* 6000 ENTRIES DISTRIBUTED TO 3 FILES 13.6178 110213 (C)
*
* YOU HAVE CREATED 6 FILES
* NAME # ENTRIES
* FYSTEMA 668
* FYSTEMM 3369
* FYSTEMO 2086
* FYSTEMT 407
* FYSTEMH 551
* FYSTEMC 415 (D)
*
* CPU SEC IN DISTRIBUTE = 17.416
* CPU SEC = 17.6856 TIME = 11:04:39
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? >stop
* TIME 017.688
* * * * *

```

Figure 4 l. Here we see how the DISTRIBUTE module divides a file into six subfiles, suitably named and saved for further processing. See the text for an explanation of how the files are named by the DISTRIBUTE module. The numbers at [C] are the cpu seconds elapsed since the start of the operation and the wall-clock time.

```

*****
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? >monitor,500,aggregate
*
*   *TYPE LABEL TO BE AGGREGATED -->? > sg
*   *TYPE NEW ENTRY, OLD ENTRIES TO BE AGGREGATED -->
*   ? >'14','p21/a.      ','p21/a      ','p21/n.      ','p21/n      '
*   ? >'14','p21/c.      ','p21/c      '
*   ? >' 4','p21.        ','p21      '
*   ? >' 9','aa.         ','cc.         ','ia.         ','ic.         '
*   ? >' 8','am.         ','cm.         ','im.         '
*   ? >'15','a2/a.       ','c2/c.       ','i21/a.      ','i2/a.      '
*   ? >'12','a2/m.       ','c2/m.       ','i2/m.       '
*   ? >' 5','a2.         ','c2.         ','i2.         '
*   ? >'15','i2/c.      '
*   ? >' 7','pa.         ','pc.         ','pn.         '
*   ? >' 6','pm.         '
*   ? >'11','p21/m.     '
*   ? >'13','p2/a.      ','p2/c.      ','p2/n      ','p2/n      '
*   ? >'10','p2/m.     '
*   ? >' 3','p2.        '
*   ? >end
*   *TYPE NAME OF NEW VECTOR OR R(EPLACE) TO OVERWRITE -->
*   ? > sgnum
*****

```

The diagram consists of four callouts labeled A, B, C, and D, each enclosed in a circle. Callout A points to the command '>monitor,500,aggregate'. Callout B points to the command '> sg'. Callout C points to the first column of the table of entries, specifically the entry '>'14','p21/a. ','p21/a ','p21/n. ','p21/n '''. Callout D points to the command '> sgnum'.

Figure 4m. See the text for a discussion of this application of the AGGREGATE module. In the degenerate case this module serves to translate data items (one-for-one). See section 5.2 for a fuller account of the features of this module.

\*\*\* NOTES \*\*\*

## 5. Descriptions of the Data Manipulation and Analysis Modules

In the previous section we have tried to convey a general view of the modular character of Omnidata and illustrate the nature of the dialogue between the user and the system. In this and the following sections we take up, in alphabetic order, the features of each of the modules. This we have done to reduce to a hopeful minimum the time spent in searching for an explanation of a particular module.

The following modules are discussed in this section:

ABRIDGE	AGGREGATE	ANALYSIS	ARRAY
BROWSE	COMPUTE	CONCAT	CROSSTAB
DESCRIBE	DISPLAY	DISTRIBUTE	ENCODE
EXTRACT	FETCH	FIT	GRAPH
KWOC	PLAN	PLOT	RANDOM
REGRESS	RENAME	REPORT	SAVE
SEARCH	SEGMENT	SEQUENCE	SORT
STACK	STATIS	STATPLOTS	SUMMARY
SURVEY	TALLY	TRIM	

The Omnidata system contains eight more modules which are of primary interest to the file builder or the data file administrator. These utility modules are described briefly in section 1.5 and more fully in section 6.

## 5.1 ABRIDGE

In a typical personnel file there may be more than 100 data items per person (see fig. 3.1e). Our experience with such a file over the last few years confirms the fact that it is sometimes necessary and often desirable to be able to search the file on any of the data items. It is, indeed, a feature of the Omnidata system that all of the data items in an Omnidata file are searchable. We recognize, nevertheless, that seldom used information in the file represents an overhead which may be burdensome in repeated uses of large files. The ABRIDGE module was designed to assist in excluding seldom used data from the working copy of the file or even in partitioning the file into two or more segments in which the data vectors are grouped for more efficient operation.

This module requests as input from the user the names of those vectors which will (or will not) be included in the abridged file. The user has the option of *selecting* certain data or of *rejecting* certain data, whichever is easier to specify. If only a few vectors are to be retained in the abridged file, the user may input 'SELECT' or 'S' followed by a list of the labels for those data vectors to be extracted. If, on the other hand, the majority of the data vectors are to be retained, it may be easier for the user to type 'REJECT' or 'R' followed by labels for data vectors to be excluded from the final file. In either case, the ABRIDGE module creates a smaller file consisting of the same number of entries, but fewer data points per entry. When typing labels for vectors to be selected or rejected, continuation of the instruction to the next line can be achieved by ending the line with the sequence: comma, ampersand, carriage return.

Vectors to be selected (or rejected) may be specified either by label or vector number and in any order. The order of vectors in the abridged file is in the order of the original file. Regardless of how specified, the labels carry their original numbers. A provision has been made in the RENAME module to resequence the label numbers. If the original file contained *check sums*, ABRIDGE will inquire if they are desired in the new file also, since computation of check sums is time consuming. When the abridged file is completed, the module switches back to Omnidata which prints out the cost statement and asks for the next module. At this stage, the abridged file is available for any of the Omnidata operations or can be catalogued permanently via the SAVE module. If a specified label is not in the file, this module ignores it, proceeds with the abridgement, and prints out "XYZ WAS IGNORED. IT IS NOT A LABEL."

```

*****
*      *WHICH DATA BASE DO YOU WANT -->? flndemo (1)
*
* FILE FLNDEMO CONTAINS 110 DATA ITEMS FOR 50 RECORDS.
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -->? no cost, abridge (2)
*
*      TYPE S(ELECT) OR R(EJECT) -->? select (3)
*      *TYPE LABELS TO BE SELECTED -->
*      ? ss , name, grade, step, yob, coldeg, & (4)
*      ? yr, yod, xyz (4)
*      YOD WAS IGNORED. IT IS NOT A LABEL. (5)
*      XYZ WAS IGNORED. IT IS NOT A LABEL. (5)
*
*      *TYPE A MODEUL NAME AND/OR INSTRUCTIONS
*      -->? labels
*      THE FILE FLENDEMO CONTAINS DATA LABELLED AS.FOLLOWS:
*      35 COLDEG      18 GRADE      5 NAME      2 SS#      19 STEP
*      7 YOB          37 YR
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS (6)
*      -->? save
*
*      *TYPE NAME UNDER WHICH FILE IS TO BE SAVED -->? demoab
*      DUPLICATE FILE NAME (7)
*      *TYPE NEW NAME OR REPLACE -->? replace (7)
*      DEMOAB IS CATALOGUED- 64 TRACKS 50 RECORDS 10.0958
*
*      *TYPE NAME UNDER WHICH FILE IS TO BE SAVED -->? end
*
*      TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -->? cost, fetch (8)
*
*      *WHICH DATA BASE DO YOU WANT? -->? demoab
*      FILE DEMOAB CONTAINS 7 DATA ITEMS FOR 50 RECORDS.
*      THE FOLLOWING LABELS ARE IN THE FILE:
*      SS , NAME, YOB, GRADE, STEP, COLDEG, YR
*
*      CPU SEC IN FETCH = 1.8462
*      CPU SEC = 14.9138      TIM = 17:07:47
*****

```

Figure 5.1a. A record of the use of the ABRIDGE module in the SELECT mode, followed by other typical module sequences. 1) This file is a small version of FNDEMO. 2) Here we suppress the cost statement which is reinstated in item 8 below. 3) Here we select only a few of the data vectors. 4) Note the ampersand for continuing on the next line. 5) The result of typing a non-existing label. 6) The labels retain their existing numbers except if resequenced by the RENAME module. 7) Here we save the abridged file to illustrate how the SAVE module handles duplicate file names. 8) Here we reinstate the cost printout and FETCH the file to confirm that the file has been abridged properly.

```

*****
* *WHICH DATA BASE DO YOU WANT- -? fsfcp
* FILE FSFCP CONTAINS 17 DATA ITEMS FOR 83 RECORDS.
*
* TYPE A MODULE NAME AND/OR INSTRUCTIONS
* - - ->? labels,no cost, abridge _____ ①
*
* THE FILE FSFCP CONTAINS DATA LABELLED AS FOLLOWS:
* 7 ADD          11 AMOUNT      16 AM72      2 CCODE     3 CINST
* 8 COUNTRY      13 DB          15 DE        10 DIV      1 NO
* 6 NAME         9 NBSNAME     4 PROGC     17 REM      5 TITLE
* 12 YB          14 YE
*
* *TYPE S(ELECT) OR R(EJECT) - - ->? reject
*
* *TYPE LABELS TO BE REJECTED - - ->?
* ? title,name,add,nbsname,rem _____ ②
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* - - ->? labels
*
* THE FILE FSFCP CONTAINS DATA LABELLED AS FOLLOWS:
* 11 AMOUNT      16 AM72      2 CCODE     3 CINST     8 COUNTRY
* 13 DB          15 DE        10 DIV      1 ID        4 PROGC
* 12 YB          14 YE
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* - - ->? rename _____ ④
*
* PRESENTLY, THE LABELS IN THE FILE ARE:
* ID, CCODE, CINST, PROGC, COUNTRY, DIV, AMOUNT, YB, DB, YE, DE,
* AM72
*
* *TYPE OLD LABEL, NEW LABEL--TYPE END TO STOP - - ->
* ? amount,cost
* ? resequence _____ ⑤
* *ENTER INTIAL VALUE AND STEP - - ->? 1.1
* ? end
*****

```

Figure 5.1b. A record of the use of the ABRIDGE module in the REJECT mode, followed by the resequencing operation of the RENAME module. 1) Note that Omnidata is still in control at this point and recognizes the instructions LABELS, NO COST, etc. 2) Here we abridge the file by rejecting certain data vectors. 3) Note that the labels retain their original numerical designations until modified in the RENAME module (item 5 below). 4) See section 5.22 for a discussion of RENAME.

## 5.2 AGGREGATE

The main purpose of this module is to allow the user of a file to classify together any of the attributes in a particular data vector into a more general category. Thus we may wish to flag as physical scientists, all chemists, physicists, metallurgists; as life scientists, all biologists, botanists, physiologists, surgeons, etc.; as engineers, all EE's, ME's, CE's, etc. A data file on repair records of automobiles provides another example where the AGGREGATE module can be put to good use. Thus, the following input:

```
GM, BUICK, CHEVY, PONTIAC, OLDS, CADILLAC
FORD, FORD, LINCOLN, MERCURY
CHRYSLER, DODGE, PLYMOUTH, CHRYSLER
```

allows one to analyze the data by manufacturer as well as by model name.

The aggregation operation can be performed in two ways. A new data vector can be added to the file or the changes can be made *in situ* by writing over the existing data vector. A typical dialogue follows:

```
*WHICH MODULE DO YOU WANT--->? > aggregate
*TYPE LABEL TO BE AGGREGATED--->? > title
*TYPE NEW ENTRY, OLD ENTRIES TO BE AGGREGATED--->? >
?physical, phys, chem, spect, spekt, eng
?life, md, biol, bot, ent
?misc, other
?end
*TYPE NAME OF NEW VECTOR, OR R(EPLACE) TO OVERWRITE--->? >
?replace
```

A word of caution is in order here. Unlike the SEARCH module—in which PHYS will match anything containing the letters PHYS—this module takes its instructions more literally. The data item in the file must match PHYS, CHEM, or ENT exactly (except for leading and trailing blanks) for the action to be taken. Provision for aggregating on the basis of string fragments in the file has been provided via the use of asterisks as discussed below.

In the above illustration, all entries with titles other than those enumerated are classified as miscellaneous. Had we left out instruction MISC, OTHER, all other titles would remain unchanged in the new vector. If a new data vector is generated, the original detailed information is still available and can be used for aggregation again at a lower or higher hierarchical level. Obviously the user must supply labels for the new vectors thus created.

As can be seen from the following figures, aggregations are performed on one data vector at a time. Figures 5.2a and 5.2b show the use of this module not to aggregate, but simply to translate the data items. Thus if a code is used for sex (1 for male and 2 for female), the instructions M,1 and F,2 will replace all of the 1's by M's and all of the

2's by F's in the SEX field. In the above instance, since one letter replaces one number, the changes are made without rearranging the file on disc. If, however, we wish to replace the 1 by the word MALE and the 2 by FEMALE, the module recognizes that the new data elements cannot be written over the old; it finds empty spaces in the assigned sectors to accommodate the longer information, rewrites the file, and resets the pointers in the LABEL FILE accordingly.

When AGGREGATE is used to generate a new data vector, the original vector is unchanged and the space allotted to the new vector is equal to the longest new entry. This poses no problem for the data that are replaced, but may present a problem with longer data items that are not modified. Items not specifically mentioned in the instructions can either be left alone (but truncated if necessary) or may be replaced by a single string such as MISC in the above example.

In the example shown earlier the instruction PHYSICAL, PHYS, CHEM, SPECT, etc. would put into the title vector the designation PHYSICAL wherever the string PHYS, CHEM, etc. appears. If we wish to make the replacement for data containing these string fragments at the beginning of the data field, we must type PHYS\*, CHEM\*, etc., to replace either PHYSICIST, or CHEMIST, etc. On the other hand, if we wish to replace ASTROPHYSICS by PHYSICAL, we need to type \*PHYS\*. To complete the picture, an asterisk in front or in back of a character string will replace all data items ending or starting with that character string, respectively.

In figure 5.2c, we show how the AGGREGATE module is used to generate a new data vector called STATE by recognizing the state abbreviation at the end of the field CITY/ST. In the process, we take into account a number of possible ways of indicating states (DC or D.C. or " ,D C", etc.).

\* \* \* N O T E S \* \* \*

```

*****
*      *WHICH DATA BASE DO YOU WANT -->?  fndemo
*
*  FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*  -->?  aggregate
*
*      *TYPE LABEL TO BE AGGREGATED -->?  sex
*      *TYPE NEW ENTRY, OLD ENTRIES TO BE AGGREGATED -->
*  ? m,1
*  ? f,2
*  ? end
*
*      *TYPE NAME OF NEW VECTOR, OR R(EPLACE) TO OVERWRITE --->
*  ? nsex
*
*  CPU SEC IN AGGREGATE = 18.363
*  CPU SEC = 22.972          TIM = 10:28:45
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*  -->?  display
*
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*  -->?  SEX, NSEX
*
*  S      N
*  2      F
*  1      M
*  1      M
*  1      M
*  1      M
*  1      M
*  1      M
*  1      M
*  1      M
*  1      M
*  1      M
*
*      *MORE -->?  no
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*  -->?  end
*****

```

Figure 5.2a. Example of the use of the AGGREGATE module to: 1) simply translate the codes 1 and 2 in the SEX field to read M and F; 2) store the results in a new vector called NSEX; 3) in order to show that the change had actually been made. Note that the DISPLAY module truncates the column heading to agree with the defined width of the field, but adds the new label NSEX to the label file. The next figure shows what happens when we replace the entries by longer ones.

```

* * * * *
*      *TYPE LABEL TO BE AGGREGATED -->? sex
*      *TYPE NEW ENTRY, OLD ENTRIES TO BE AGGREGATED -->
*
*      male,1
*      ? female,2
*      ? end
*      *TYPE NAME OF NEW VECTOR, OR R(EPLACE) TO OVERWRITE--->
*      ? nisex
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -->? display
*
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*      -->? sex,nsex,nisex
*
*      S      N      NISEX
*      2      F      FEMALE
*      1      M      MALE
*      *MORE -->? yes
*      1      M      MALE
*      2      F      FEMALE
*      1      M      MALE
*      *MORE -->? no
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*      -->? end
* * * * *

```

Figure 5.2b. Here we generate still another data vector in which the words MALE and FEMALE are written in place of 1 and 2. Note in the DISPLAY of the vectors SEX, NSEX, and NISEX how the column headings are handled.

```

*****
*          *WHICH DATA BASE DO YOU WANT -->? fcarpool
*
* FILE FCARPOOL CONTAINS 35 DATA ITEMS FOR 500 RECORDS.
*
*          *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? no cost,aggregate
*
*          *TYPE LABEL TO BE AGGREGATED -->? city/st
*          *TYPE NEW ENTRY, OLD ENTRIES TO BE AGGREGATED -->
* ? maryland,* md*,*mary*
* ? virginia,* va*,*virg*
* ? dc,* dc*,*d c*,*d.c.*,*d. c.*
* ? illinois,*ill*,*il*
* ? massachusetts,* mass*,*ma*
* ? colorado,*colo*,*co *
* ? florida,*fa*,*fla*,*flor*
* ? indiana,*ind*,*in *
* ? california,*ca*,*calif*
* ? delaware,*del*
* ? other,other
* ? end
*          *TYPE NAME OF NEW VECTOR, OR R(EPLACE) TO OVERWRITE--->
* ? state
*
*          *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? display
*
*          *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* -->? city/st,state
*
*          CITY/ST
*          LANTZ MD
*          WALKERVILLE MD
*          WILMINGTON DEL
*          GERMANTOWN MD
*          ROCKVILLE MD
*          SILVER SPRING MD
*          SUNNYVALE CA
*          BELTSVILLE MD
*          HOUSTON TX
*
*          STATE
*          MARYLAND
*          MARYLAND
*          DELAWARE
*          MARYLAND
*          MARYLAND
*          MARYLAND
*          CALIFORNIA
*          MARYLAND
*          OTHER
*****

```

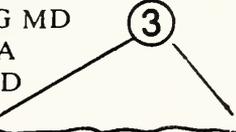
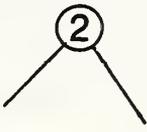
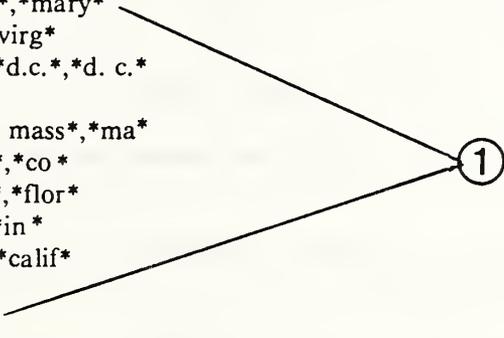


Figure 5.2c. Here we use the main facility of the AGGREGATE module to generate a data vector called STATE on the basis of information in the vector CITY/ST. Note: 1) we can allow for misspellings or other variations; 2) how the state designation was extracted from the city field; and 3) how states not mentioned specifically are labeled as OTHER.

```

* * * * *
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -- ->? aggregate
*
*      *TYPE LABEL TO BE AGGREGATED -- ->? state
*      *TYPE NEW ENTRY, OLD ENTRIES TO BE AGGREGATED -- ->?
*      ? east,mary*,virg*,mass*,dc*,del*
*      ? south,flor*
*      ? west,cal*
*      ? central,col*
*      ? midwest,ill*,ind*
*      ? other,other
*      ? end
*      *TYPE NAME OF NEW VECTOR, OR R(EPLACE) TO OVERWRITE --->
*      ? region
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -- ->? display
*
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*      -- ->? city/st,state,region
*
*      CITY/ST                STATE                REGION
*      LANTZ MD                MARYLAND                EAST
*      WALKERVILL MD           MARYLAND                EAST
*      WILMINGTON DEL          DELAWARE                EAST
*      GERMANTOWN MD           MARYLAND                EAST
*      ROCKVILLE MD           MARYLAND                EAST
*      SILVER SPRING MD        MARYLAND                EAST
*      SUNNYVALE CA            CALIFORNIA              WEST
*      BELTSVILLE MD          MARYLAND                EAST
*      GAITHERSBURG MD          MARYLAND                EAST
* * * * *

```

Figure 5.2d. Here we operate on the STATE vector to AGGREGATE states into geographic regions.

### 5.3 ANALYSIS

This module operates on the designated numerical data vectors to compute and print out the maximum value, minimum value, total value, and average value for each of the specified data vectors. Also printed are the labels for the vectors and the number of items encountered in each vector. This latter figure serves to indicate how many items have been used in the total and average computations and serves also to alert the user to the fact that data items are missing in certain vectors. Zeros are considered valid data items; blanks are not, nor are alphabetic or graphics. A question mark is printed in the verify column to draw attention to missing data items. A number of utility modules are available to assist in screening and updating of the missing items.

Where it is possible to update or otherwise correct for the missing data items, Omnidata provides a SCREEN module to assist in that operation and an UPDATE module to correct the file.

The ANALYSIS module operates also in the *tandem mode*, wherein the desired analysis is performed in sequence on more than one file. In the TANDEM mode, the module looks first for the scratch file FFILES which contains the names of subfiles created by a DISTRIBUTE operation during the current run. If such a file exists, the module prints the names of the files before it carries out the analysis to make sure that these are indeed wanted. If FFILES does not exist (meaning that no files have been distributed in the current run), the module asks the user to supply names of these cataloged files he wishes to analyze in tandem. Missing files in the list supplied will be ignored as will data vectors which may be missing in one or more of the designated files. Thus, the files need not all be identical in content. All that is required is that they be in Omnidata format. Figures 5.3a, 5.3b, and 5.3c show representative operations in ANALYSIS and ANALYSIS, TANDEM.

If an analysis is desired for all of the numeric data vectors, it can be obtained more expeditiously by calling the SUMMARY module. There the analysis is the same; there is no need to specify the data vectors. They are all summarized automatically as shown in figure 5.32a.

We now illustrate the use of the ANALYSIS module on a sample personnel file called FDEMO containing information for 350 persons distributed among 5 regions. We have used the same file to illustrate the applications of other modules (CROSSTAB, STATIS, SUMMARY, and TALLY) in order to emphasize the various levels of data analysis available in the Omnidata package.

In figure 5.3a we see the result of the analysis of 5 numeric data vectors in a file of 500 persons. The numbers in the total column are meaningful only for the PAY item. The question mark in the VERIFY column alerts us that the total and average values are based on less than 500 records. In this case only 394 records have numerics in the degree field. The remainder probably contain blanks.

If we wish to perform a similar analysis separately on each of the five regional offices represented in this file, the ANALYSIS module facilitates this by allowing a TANDEM mode as mentioned earlier. Figure 5.3b shows how this is achieved. There we did an analysis on only three data items so as to compress the printout.

```

*****
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? >analysis
*
*
*
*
*   *TYPE LABELS TO BE ANALYZED --> >job,level,pay,eody,deg
*
*
*
*
* LABEL      ITEMS    MAX     MIN     TOTAL      AVERAGE  VERIFY
* -----
* YOB        500      57      4       16332      32.664
* LEVEL      500      17      0       4621       9.242
* PAY        500      36000   0       7.499638E+6  14999.3
* EODY       500      73      35      32100      64.2
* DEG        394      4       0       520        1.31980   ?
*
*
*
*   *TYPE LABELS TO BE ANALYZED --> >end
*****

```

Figure 5.3a. These results from the ANALYSIS module agree with those from the SUMMARY module (see fig. 5.32a). If the file carried a current summary, this module would get its information there instead of computing it again.

```

*****
*      *TYPE LABELS TO BE ANALYZED --->? end
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? distribute
*
*      *DISTRIBUTE ON WHICH LABEL --->? region
* YOU HAVE CREATED 5 FILES
* NAME          # ENTRIES
* FIREGION3     72
* FIREGION2     175
* FIREGION6     26
* FIREGION1     132
* FIREGION4     95
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? analysis,tandem
*
* THE FOLLOWING LABELS ARE IN THE FILE:
* REGION,SS #,SEX,CIT,NAME,DOB,YOB,SCD,STAT,TOA,TOD,TOLA,NTE,PP,
* OSC,FC,TITLE,GRADE,STEP,SCALE,DIV,SEC,LC,CSDFROM,CSDDDAY,
* DDGR,PINFO,DOPP,EOD,EODY,ADTIT,PROFSTAT,DEG,UNIV,YR,SPP,DPG,ATD,
* SEQ,VP,INS,HC,RET,NOAC,SLIM,DLIM,HLIM,EDA,AUT,POS #,AC,PFC,APPOR,
* REM,AUDAT,SON,SPF,PD,DRPB,CL,NTE,ANU,TP,DET,MC,ALC,SCD,GLOC,DAA,
* RS,SSNC,PF,HBPN,DPSQ,SP,GDF,SC,DLABR,SED,HWD,WAEPL,WAEDWS,RHWSW,
* WAEDWP,WAEDWT,LWOP,CNHS,TLWOP,CLWOP,CAWOL,CSL,FAB,WOD,WDUD,WED,
* AA,NP,WHSA,LWOPSA,APAD,FUD,DDED,MR,SK,HWWAE,SCDY,SALBAS,C,LEVEL,
* IDN,AGE,ID
*
*      *TYPE LABELS TO BE ANALYZED --->? age,level,eody
*****

```

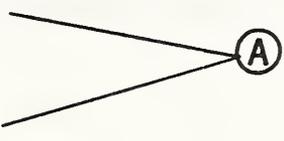


Figure 5.3b. After the operation shown in figure 5.3a the file was distributed on region so as to carry on an analysis on each region separately. See section 5.11 for an explanation of the material at [A]. In the TANDEM mode, we are reminded at [B] which data items are in the file. The result of the request at [C] is shown in the next figure.

```

*****
* THE FOLLOWING LABELS ARE IN THE FILE:
* REGION,SS #,SEX,CIT,NAME,DOB,YOB,SCD,STAT,TOA,TOD,TOLA,NTE,PP,
* OSC,FC,TITLE,GRADE,STEP,SCALE,DIV,SEC,LC,CSDFROM,CSDDDAY,
* DDGR,PINFO,DOPP,EOD,EODY,ADTIT,PROFSTAT,DEG,UNIV,YR,SPP,DPG,ATD,
* SEQ,VP,INS,HC,RET,NOAC,SLIM,DLIM,HLIM,EDA,AUT,POS #,AC,PFC,APPOR,
* REM,AUDAT,SON,SPF,PD,DRPB,CL,NTE,ANU,TP,DET,MC,ALC,SCD,GLOC,DAA,
* RS,SSNC,PF,HBPN,DPSQ,SP,GDF,SC,DLABR,SED,HWD,WAEPL,WAEDWS,RHWSW,
* WAEDWP,WAEDWT,LWOP,CNHS,TLWOP,CLWOP,CAWOL,CSL,FAB,WOD,WDUD,WED,
* AA,NP,WHSA,LWOPSA,APAD,FUD,DDED,MR,SK,HWWAE,SCDY,SALBAS,C,LEVEL,
* IDN,AGE,ID
*
* *TYPE LABELS TO BE ANALYZED --->? age,level,eody
*
* LABEL ITEMS MAX MIN TOTAL AVERAGE VERIFY
* -----
* 1 AGE 132 72 21 5918 44.8333
* 1 LEVEL 132 16 0 944 7.15152
* 1 EODY 132 73 40 8754 66.3182
*
* 2 AGE 175 73 22 7817 44.6686
* 2 LEVEL 175 17 0 1726 9.86286
* 2 EODY 175 73 35 10994 62.8229
*
* 3 AGE 72 66 20 3177 44.125
* 3 LEVEL 72 17 0 736 10.2222
* 3 EODY 72 73 40 4482 62.25
*
* 4 AGE 95 68 20 4061 42.7474
* 4 LEVEL 95 16 0 938 9.87368
* 4 EODY 95 73 41 6199 65.2526
*
* 5 AGE 26 71 23 1195 45.9615
* 5 LEVEL 26 16 0 277 10.6538
* 5 EODY 26 73 40 1671 64.2692
*
* TRANSLATION TABLE FOR FILES
* 1 = FIREGION1
* 2 = FIREGION2
* 3 = FIREGION3
* 4 = FIREGION4
* 5 = FIREGION6
*****

```

Figure 5.3c. Here we see how the tandem option formats the analysis. The files are numbered in the table to conserve space, but the names of the files are listed below.

## 5.4 ARRAY

The utility of a comprehensive data retrieval or analysis package is often enhanced when it can accept data from or deliver data to other programs or systems. This module was designed to write all the numeric data vectors from an Omnidata file into a separate cataloged file in proper format for the OMNITAB II system to handle it.

The OMNITAB II system is limited to 12,500 data points and has a worksheet normally dimensioned for 201 rows by 62 columns. The ARRAY module instructs the users that the array it has created fits into the normal OMNITAB work sheet (see fig. 5.4a) or how to change the dimension of the OMNITAB II work sheet to accommodate the array (see fig. 5.4b). If the array exceeds the 12,500 limit, this module instructs users how to segment the file. Figure 5.4c shows a record of the instructions supplied in this instance. Since this module stores all numeric vectors, we called the ABRIDGE module first to select only those data vectors we wished to transfer to another program or system.

```
* * * * *
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -->? array
*      YOU HAVE CREATED THE FILE ARRAYNDEMO WITH 50 ROWS
*      THE 4 COLUMNS ARE:
*      1 . YOB (7)
*      2 . GRADE (18)
*      3 . PAY (20)
*      4 . EODY (32)
*      THE ARRAY IS 50 BY 4 --
*      A TOTAL OF 200 ENTRIES
*      WHEN READING ARRAYNDEMO INTO OMNITAB OR ANY OTHER PROGRAM.
*      SKIP THE FIRST 5 HEADER RECORDS.
*      YOUR FORMAT FOR READING IN THE DATA IS AS FOLLOWS:
*      4F12.0
*      THE OMNITAB WORKSHEET IS AUTOMATICALLY DEFINED AS
*      201 ROWS BY 62 COLUMNS
*      YIELDING ROOM FOR A MAXIMUM OF 12500 ENTRIES.
*      YOUR WORKSHEET CAN STAY DEFINED AS IS.
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* * * * *
```

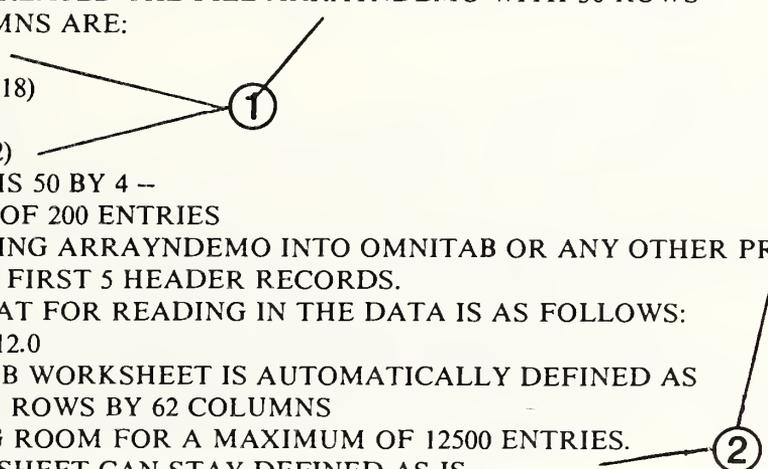


Figure 5.4a. Having selected 50 records at RANDOM from a larger file FNDEMO and then used ABRIDGE, we entered the ARRAY module with a file containing seven data vectors, of which only four were numeric. This output shows: 1) how the resulting file is named and described in terms of the labels and their numbers in the original file; and 2) the output when the array of data fits into the normally dimensioned worksheet of OMNITAB II. The next figure shows the output when the array exceeds 201 records.

```

*****
*           *WHICH DATA BASE DO YOU WANT -->?  fndemo
*
* FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->?  no cost,abridge
*
* *TYPE S(ELECT) OR R(EJECT) -->?  s
* *TYPE LABELS TO BE SELECTED -->
* ? ss#,name,pay,grade,step,eody,youb,scdy,deg,yr,title
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->?  array
*
* YOU HAVE CREATED THE FILE ARRAYNDEMO WITH 500 ROWS
* THE 8 COLUMNS ARE:
*   1 . YOY (7)
*   2 . GRADE (18)
*   3 . STEP (9)
*   4 . PAY (20)
*   5 . EODY (32)
*   6 . DEG (35)
*   7 . YR (37)
*   8 . SCDY (108)
* THE ARRAY IS 500 BY 8 --
*   A TOTAL OF 4000 ENTRIES
* WHEN READING ARRAYNDEMO INTO OMNITAB OR ANY OTHER PROGRAM.
* SKIP THE FIRST 9 HEADER RECORDS.
* YOUR FORMAT FOR READING IN THE DATA IS AS FOLLOWS:
*   6F12.0/2F12.0
* THE OMNITAB WORKSHEET IS AUTOMATICALLY DEFINED AS
*   201 ROWS BY 62 COLUMNS
*   YIELDING ROOM FOR A MAXIMUM OF 12500 ENTRIES.
* YOUR WORKSHEET MUST BE REDEFINED TO HAVE AT LEAST
*   500 ROWS. THEREFORE ALLOWING FOR A MAXIMUM OF
*   25 COLUMNS.
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->?  stop
*****

```

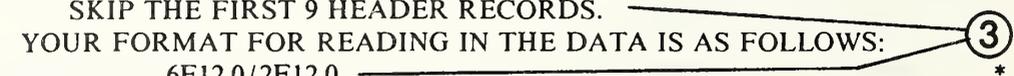
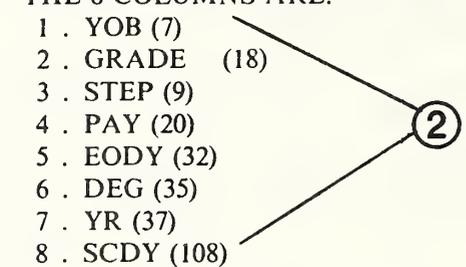
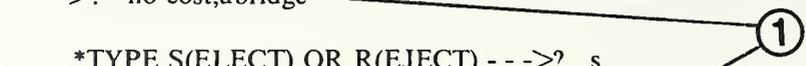


Figure 5.4b. Here we ABRIDGE 1) a file to contain 11 data vectors of which 8 are numeric. Note: 2) the names of the numeric vectors and their numbers in the original file; 3) format instructions; and 4) advice to redimension the OMNITAB II worksheet. The next figure shows the nature of the instructions for partitioning the file when it exceeds 12,500 data points. See fig. 5.4d for the array produced by this operation and how an OMNITAB II run would make use of it.

```

* * * * *
*   YOU HAVE CREATED THE FILE ARRANYNDEMO WITH 500 ROWS   *
*   THE 36 COLUMNS ARE:                                     *
*   1 . DIV (1)                                             *
*   2 . YOB (7)                                             *
*   3 . FC (16)                                             *
*   4 . GRADE (18)                                          *
*   5 . STEP (19)                                           *
*   6 . PAY (20)                                            *
*   7 . PLANT (21)                                          *
*   8 . DEPT (22)                                           *
*   9 . EODY (32)                                           *
*   10 . DEG (35)                                           *
*   11 . YR (37)                                            *
*   12 . VP (42)                                            *
*   13 . INS (43)                                           *
*   14 . HC (44)                                            *
*   15 . RET (45)                                           *
*   16 . NOAC (46)                                          *
*   17 . PFC (54)                                           *
*
*   29 . NP (99)                                            *
*   30 . NHSA (100)                                         *
*   31 . LWOPSA (101)                                       *
*   32 . MR (105)                                          *
*   33 . SK (106)                                          *
*   34 . HWWAE (107)                                       *
*   35 . SCDY (108)                                        *
*   36 . C (110)                                           *
*
*   THE ARRAY IS 500 BY 36 --                                *
*   A TOTAL OF 18000 ENTRIES                                *
*   WHEN READING ARRAYNDEMO INTO OMNITAB OR ANY OTHER PROGRAM.
*   SKIP THE FIRST 37 HEADER RECORDS.                       *
*   YOUR FORMAT FOR READING IN THE DATA IS AS FOLLOWS:   *
*       6F12.0/6F12.0/6F12.0/6F12.0/6F12.0/6F12.0        *
*   THE OMNITAB WORKSHEET IS AUTOMATICALLY DEFINED AS     *
*       201 ROWS BY 62 COLUMNS                             *
*       YIELDING ROOM FOR A MAXIMUM OF 12500 ENTRIES.     *
*   YOUR FILE IS TOO LARGE FOR OMNITAB TO HANDLE.         *
*       YOU NEED 36 COLUMNS--PLEASE CALL SEGMENT        *
*       TO SEPARATE ARRAYNDEMO INTO SUBFILES OF NOT MORE THAN
*       347 ROWS EACH.
* * * * *

```

Figure 5.4c. In this run ARRAY assumes that the file must contain all of the 36 data vectors and instructs the user how to SEGMENT the file to a maximum of 347 records in order to fit into a suitably defined OMNITAB II worksheet.



## 5.5 BROWSE

The BROWSE module is designed to familiarize a user with the information in the data file. When the user specifies a label on which he wishes to browse, this module reads through the file and prints the unique entries in the specified vector. Thus, if the user wished to browse on 'TITLE,' entries such as 'CHEMISTS,' 'PHYSICIST,' 'MATHEMATICIAN,' etc., would be printed. After 10 unique titles have been printed, the user is asked if he wishes to see more. If he doesn't need to browse any longer on that label, he may select another label to browse on, or he may enter 'END' or 'NONE' and be transferred back to Omnidata. See figure 5.5a for a typical use of this module.

The response to the question MORE? can be either YES or NO or a number. BROWSE prints out only unique entries. When 100 records are read without finding a new item to be printed, this module prints:

```
XX% OF THE FILE READ. NO NEW HITS IN THE LAST 100 RECORDS
* MORE --->? >
```

Thus, the user has the option to continue or END.

BROWSE keeps a running tally of the frequency of occurrence of each of the unique items it finds. Before returning to Omnidata, the user is asked whether a frequency distribution of the displayed items is desired.

When the *monitor switch* is on, the module prints also the number of the first record in the file in which the particular entry appeared.

As described above, the BROWSE module performs its function by reading the actual file. The cost in computer time to do this depends in large measure upon the variety of entries in the vector in question. If a vector listing the handicap code contains simply the entries 1, 2, 3, 4, it will take much less time to browse through the file on that vector than upon the vector containing occupations that may run to 200 or 300 titles. The BROWSE module will have its primary utility just in exploring the way information is entered in the file. It can quickly answer the question whether citizenship is designated by a number code or by listing the country, or whether sex is indicated by M and F or by 1 and 2, etc.

When Omnidata is in productive use, BROWSE can also be used to screen the file for inadvertent errors that are not easily picked up by the SCREEN module to be described later. When one browses on job titles, for example, it is easy to pick up inconsistencies in the use of punctuation, brackets where parentheses are required, or extra spaces, etc.

Where the use of the BROWSE module on the same file is frequent, and especially if it is performed on a data vector with many variants, it would be more economical to write the results out on a scratch file.



```

*****
*          *WHICH LABEL DO YOU WANT TO BROWSE ON -->? yob *
*  LINE    REC NO.  YOBB *
*  -----  -----  ---- *
*  1.      1        37 *
*  2.      2        15 *
*  3.      3        22 *
*  4.      6        28 *
*  5.      8        25 *
*  6.      9        29 *
*  7.     11        42 *
*  8.     12        19 *
*  9.     13        38 *
* 10.     14        21 *      *MORE -->? 25
* 11.     15        34 *
* 12.     16        36 *
*-----*
* 34.     67        18 *
* 35.     75        23 *      *MORE -->? 25
* 36.     81        45 *
*-----*
* 52.     450       11 *
* 53.     495       57 *
*  END OF FILE REACHED 500 RECORDS READ *
*  *DO YOU WANT A TALLY PRINTED OUT -->? yes *
*  1.     20        15.    13        29.    10        43.    2 *
*  2.     7         16.    10        30.     9        44.    3 *
*  3.    18         17.     9        31.    11        45.    6 *
*  4.    18         18.    22        32.    12        46.    2 *
*  5.    18         19.     2        33.     9        47.   12 *
*  6.    16         20.    14        34.     4        48.     5 *
*  7.     4         21.     9        35.     9        49.     1 *
*  8.    13         22.    13        36.    13        50.     1 *
*  9.     8         23.    14        37.     7        51.     2 *
* 10.    13         24.     6        38.    11        52.     1 *
* 11.    10         25.    12        39.    11        53.     2 *
* 12.     9         26.    13        40.    11 *
* 13.    10         27.     5        41.     7 *
* 14.     9         28.    11        42.    13 *
*  FOR HISTOGRAMS, CALL TALLY *
*  *WHICH LABEL DO YOU WANT TO BROWSE ON -->? stop *
*  PROGRAM STOPPED. *
*****

```

Figure 5.5b. A typical use of BROWSE with the MONITOR set very high showing: 1) 25 entries printed at a time; and 2) the table showing frequency of occurrence of each of the displayed data.

```

*****
*          *WHICH LABEL DO YOU WANT TO BROWSE ON -->? city/st *
*  LINE   CITY/ST *
*  1.     LANTZ MD *
*  2.     WALKERVILL MD *
*  3.     WILMINGTON DEL *
*  4.     GERMANTOWN MD *
*  5.     ROCKVILLE MD *
*  6.     SILVER SPRING MD *
*  7.     SUNNYVALE CA *
*  8.     BELTSVILLE MD *
*  9.     GAITHERSBURG MD *
*  10.    NEW CARROLLTON MD *MORE -->? 25 *
*  11.    WASHINGTON DC *
*  12.    CHEVERLY MD *
*-----*
*  34.    WASH GROVE MD *
*  35.    CABIN JOHN MD *MORE -->? yes *
*  36.    WASHINGTON DC *
*  37.    MANASSAS VA *
*  38.    FAIRFAX VA *
*  39.    HYATTSVILLE MD *
*  40.    JEFFERSON MD *
*  41.    MONORIVA MD *
*  42.    IJAMSVILE MD *
*  43.    BROOKEVILLE MD *
*  44.    ASHTON MD *
*  45.    ALEXANDRIA VA *MORE -->? no *
*  *DO YOU WANT A TALLY PRINTED OUT -->? yes *
*  1.     1          13.    1          25.    1          37.    }
*  2.     1          14.    1          26.    1          38.    }
*  3.     1          15.    1          27.    8          39.    }
*  4.     2          16.    2          28.    1          40.    }
*  5.     13         17.    3          29.    1          41.    }
*  6.     13         18.    1          30.    1          42.    }
*  7.     1          19.    3          31.    1          43.    }
*  8.     1          20.    2          32.    2          44.    }
*  9.     35         21.    3          33.    1          45.    }
*  10.    2          22.    1          34.    1
*  11.    13         23.    2          35.    1
*  12.    1          24.    1          36.    1
*  FOR HISTOGRAMS, CALL TALLY *
*
*          WHICH LABEL DO YOU WANT TO BROWSE ON -->? end *
*****

```

Figure 5.5c. A use of the BROWSE module showing the variation in output produced when the MONITOR is turned off.

## 5.6 COMPUTE

In the COMPUTE module, the user has the option of creating new data vectors for each entry (record) in the file by performing simple arithmetic operations on existing numeric vectors.

When this module is selected it requests the user to:

'INPUT NEW LABEL AND DEFINITION'

The user responds with a simple equation such as

```
AGE      =      77      -      YOB
%        =      FY76    /      TOTALFY76    *      100.
```

```
!COST    =      .5      *      L2      +      L3
```

OR

```
!COST    equals    .5      TIMES    L2      PLUS    L3
```

```
TOTAL    =      FY1      +      FY2      +      FY3
```

```
NEWCOST  =      !COST    *      1.37
```

If the expression is acceptable, the module asks for another expression. On any such pass through the COMPUTE module the user may continue to input such equations until he finishes with 'END'. The first item in the equation becomes the name (label) for the new vector. If the first character is an exclamation point (!), the new vector defined by the expression on the right will not be added permanently to the file. If the exclamation point appears, the new vector is temporary and will only be available while the COMPUTE module is in control for use in subsequent calculations as is shown in the expression

```
NEWCOST  =      !COST    *      1.37.
```

The left-to-right scan of the equations requires the use of at least two spaces between labels and symbols. After finding the new label and the '=' or the word 'EQUALS', the arithmetic is performed by a simple left-to-right scan of the equation. The arithmetic operators may be entered in various ways: 'PLUS', 'MINUS', 'TIMES', 'OVER', 'EXP', or as '+', '-', '\*', '/', '\*\*'. The operations are performed on values in the permanent vectors, temporary vectors created by previous arithmetic expressions, or constants supplied on the right side of the equation.

If the file contains a current *summary*, the values such as TOTAL XYZ, AVERAGE ABC, MAXIMUM XYZ, MINIMUM XYZ, and ITEMS are available for use in the arithmetic expression. If the file has not been summarized, this module rejects the use of these expressions. ABC, XYZ, etc. in the above are the proper labels for the file in question. Typical items would be AVERAGE AGE, MAXIMUM SALARY, TOTAL EARNINGS, etc. Numbers may be used instead of labels if they are preceded by the letter L. Thus L6 can be used instead of DOB in an arithmetic statement in the COMPUTE module.

Complicated arithmetic evaluations which cannot be handled by a single equation due to the limited scan, can usually be handled by making use of temporary labels and dividing the operation into several smaller tasks. When the file and the list of labels have been suitably updated with the new computed information, control is returned to OMNIDATA.

If a personnel file were to contain starting salary as well as present salary, it would be possible to compute a growth factor as follows:

```
! SERVICE = 77 MINUS EODY
GFACTOR = SALARY MINUS INITIAL OVER !SERVICE
```

and a projection factor depending upon years to retirement thus:

```
!MORE YRS = 35 MINUS !SERVICE
PROSPECTS = GFACTOR TIMES !MORE YRS
```

As the above instructions are written, this module will create new data vectors which can accommodate up to 13 characters. These are entered right adjusted. If the number exceeds 8 digits, it is written in E notation (2.469282E+18, -2.829640E-10, etc.). When one is sure that the result of an arithmetic operation will be a small number it is possible to instruct this module to shorten the width of the new data field. Thus if we wished to allocate only two characters to the field in which we compute and store the length of service in years, we would type

```
SERVICE = 77 - EODY, 2
```

Allowing the user to specify the width of the computed fields produces a number of benefits:

- a) it often removes the need for adding new sectors to the file;
- b) it makes the output from DISPLAY and REPORT more compact; and,
- c) it speeds up the SEARCH operation.

Nevertheless, this feature should be used with caution as this module will truncate results from the right hand side if the field has been made too narrow. The effect of the resulting truncation is not too serious in certain instances if the number 45.3743 is written as 45.37, but it is disastrous if the number 1.25E+5 is entered as 1.25E+.

In data vectors computed by this module, it is likely that the decimal points will not line up. A provision has therefore been included in this module to perform an alignment operation.

Thus

```
ALIGN, ABC
```

would cause the module to request:

```
TYPE NEW LABEL, PICTURE ---> ?
```

If the user responds with

ABCD, xx.xxx

the module will take data from the original field ABC and rewrite it in a new field ABCD which is six characters wide—with two digits on the left of the decimal point and three on the right. If an entry in the original vector ABC were 105.3, it would get entered as \*5.3. This state of affairs should obviously be avoided and can indeed be avoided by allowing more room to the left in the above picture. It should be clear from the above that if the ALIGN operation is to be foolproof, it is important to know how large the numbers are in the original vector in order to allow enough room in the new one. This information can be gotten either from the summary if it exists, or from the SCREEN module. If, after all of these precautions, the ALIGN operation is still frustrated, it places an asterisk in the offending item, keeps a count of the number of such items and notifies the user how many such instances were encountered.

## 5.7 CONCAT

In CONCAT the user may create new vectors of data by concatenation of two or more existing data vectors. The instructions for this operation consist of a series of labels separated by commas. The first is the label (name) of the new vector to be generated by a juxtaposition of the specified vectors that follow.

There are several options in this module which need to be described. First of all, along with the concatenation of existing data points one may insert a constant string by entering this constant preceded and followed by a #. For example

ADD, #ADDRESS,NUM,STREET,CITY,STATE,ZIP

would create a new data vector of information for each entry in the file. This vector would be labeled ADD and would consist of first the word 'ADDRESS' followed by the information in each of the subsequent labels.

An interesting application of this module arises when it becomes necessary to rearrange a date field to make it amenable for sorting. If, for example, we listed a date as day month year (220973) and we wished to generate a new vector to read 730922, the operation could be accomplished in two steps. The EXTRACT module (see sec. 5.13) can be used to generate the vectors called DAY, MONTH, YEAR and then the CONCAT module can be used to generate a new vector called DATE, as follows

DATE, YEAR, MONTH, DAY

In figure 5.7a, we see an application of the CONCAT module to combine, into a single grid location, the 2 high order digits from each of two 5-digit numbers representing X and Y map coordinates.

```

* * * * *
*
*      *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
*  --> ? x-crd,y-crd,x1-crd,y1-crd,x2-crd,y2-crd
*
*      X-C      Y-C      X1-C      Y1-C      X2-CR      Y2-CR
*      ----      ----      -
*      10       16       105       164       1056       1648
*      4        24       40        240       409        2402
*      12       9        105
*
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*  -->? monitor,250,concat
*
*      *INPUT NEW LABEL AND LABELS TO BE CONCATENATED -->
*  ? xy,x-crd,y-crd
*  ? end
*  250 RECORDS COMPLETED 11.0022
*  500 RECORDS COMPLETED 12.824
*  750 RECORDS COMPLETED 14.6268
*  1000 RECORDS COMPLETED 16.482
*  1250 RECORDS COMPLETED 18.3874
*
*      *WHICH LABEL DO YOU WANT TO BROWSE ON-->? xy
*  NO.      REC NO.  XY
*  ----      -
*  1.        1        10 16
*  2.        3        4 24
*  3.        4        12 9
*  4.        5        14 10
*  5.        6        11 26
*  6.        7        13 13
*  7.        8        13 9
*  8.        9        11 16
*  9.       10        15 12
*  10       11        12 13
*  17.      21        8 7
*  18.      22        6 10
*  19.      23        999999
*  20.      24        9 16
*
* * * * *

```

Figure 5.7a. In the top section, we use the DISPLAY module to show, in the first two columns, the two leading digits from the last two columns. In the middle section, we show the instructions to concatenate X-C and Y-C into a single vector called XY. In the lower portion, we see via the BROWSE module that the concatenation has indeed been achieved. The reason for the space between the figures is the fact that the original fields were defined as containing 3 digits, as is made clear in line 19. Section 5.13 shows how the vectors X-C and Y-C were extracted from the vectors X2-CR and Y2-CR.

## 5.8 CROSSTAB

The CROSSTAB module accepts two labels and produces a two-dimensional array of the frequencies of occurrence of entries under the first label as a function of the entries under the second. It is a detailed breakdown of the contributions to the total frequency distribution for the first label produced by the TALLY module. This is a very useful tool when operating on coded files or when used in conjunction with the ENCODE module for handling survey data. Thus, the instruction to cross-tabulate GRADE by SEX gives a frequency distribution of pay grades by sex and by total staff.

After the cross-tabulation has been completed, but before it is printed, the user is asked to indicate how he wants his output. He may input a '1' for the actual values of the cross-tabulation, a '2' for the percentage of the total in the file, a '3' for the percentage of the total in the column, or a '4' for the percentage of the total in the row. There is also an option for supplying heading and footnote lines. If these are not desired, respond with a simple 'NO'. The output is then printed allowing at least seven columns to a page. On the last page of output, totals are given.

At this point, the user may respond with another option number in order to receive another form of output—again, either the actual values or one of the three types of percentages are possible. If none of these are desired, a '0' (zero) response is in order.

A cross-tabulation of AGE by GRADE will contain many columns of numbers and necessitate formatting the output on two or more pages. Alternatively, a cross-tabulation of GRADE by AGE would contain even more columns. The CROSSTAB module also produces a frequency distribution for the totals, which it prints automatically.

If, after seeing the cross-tabulation, the user wishes to generate histograms for any one or all of the vectors, he can do so. He also has the option of storing the table away on a scratch file for later use.

After a frequency distribution table is presented, the module asks if histograms are desired for any of the columns in the table. If the answer is yes, the column headings are again displayed and the user is asked to indicate for which columns he wishes histograms. The histograms that are printed in response to the instructions are individually normalized (scaled up or down) to fit the page. If the user wants histograms for all of the columns in the array, he responds with 'ALL', in which case the program asks whether they should be normalized individually or uniformly. Uniform normalization is obviously advantageous if comparisons are to be made between the histograms.

After each frequency table has been processed in the above manner, the module inquires whether a transposed table is desired. If the original table was GRADE VS AGE, the transpose would display AGE VS GRADE. In the former case, each row represents a grade while each column represents an age group. In the latter case, the rows and columns are interchanged. If the answer is 'YES', the transposed array is printed and the earlier dialogue is repeated. It should be noted that the second array is achieved by a matrix transposition and not from another CROSSTAB operation on the file. Figures 5.8a et seq. show typical features of this module.

\* \* \* N O T E S \* \* \*

\* \* \* \* \*  
 \* TYPE A MODULE NAME AND/OR INSTRUCTIONS \*  
 \* -->crosstab \*  
 \* \* \* \* \*

\* TYPE LABELS TO BE CROSSTABBED -->szip,firm \*  
 \* RESPOND WITH AN OPTION NUMBER. \*  
 \* 1 ACTUAL VALUE \*  
 \* 2 % OF TOTAL IN FILE \*  
 \* 3 % OF TOTAL IN SZIP \*  
 \* 4 % OF TOTAL IN FIRM \*  
 \* 0 TO STOP \*  
 \* -->1 \*  
 \* ANY HEADING -->company staff distribution by zip \*  
 \* ANY FOOTNOTE -->szip are three leading digits only \*  
 \* \* \* \* \*

COMPANY STAFF DISTRIBUTION BY ZIP

\* \* \* \* FIRM \* \* \*

SZIP	MISC	BENDER	ADMINI	CHESTE	DART A	FACILI	EASY M
100	0	0	0	1	0	0	0
172	0	0	1	0	0	0	0
198	0	1	0	0	0	0	0
200	21	3	35	13	14	5	5
204	0	1	0	0	0	0	0
207	40	14	27	18	45	9	7
208	22	8	11	12	19	3	4
209	9	4	6	9	6	1	1
210	2	0	1	0	1	1	0
217	18	7	19	7	9	11	7
220	3	3	2	1	3	1	1
221	3	0	0	1	1	1	0
222	1	0	1	0	2	0	0
606	0	1	0	0	1	0	0
612	0	0	1	0	0	0	0
803	0	0	0	0	1	0	0
940	1	0	0	0	0	0	0
TOTALS	121	43	107	63	105	32	25

SZIP ARE THREE LEADING DIGITS ONLY \*

THIS TABLE SHOWS ACTUAL VALUES \*

Figure 5.8a. This and the succeeding figures were run from a PLAN (see sec. 5.18) rather than from the keyboard so that the question mark following the arrow (--->) is missing. See the next figure for a continuation of this cross-tabulation and for a translation of the truncated column headings.

\*NOTE: 4 DATA POINTS WERE IGNORED.  
ALPHA DATA IN NUMERIC FIELDS: SZIP

SZIP	*** FIRM *** TOTAL
100	1
172	1
198	1
200	96
204	1
207	160
208	76
209	36
210	5
217	78
220	14
<hr style="border-top: 1px dashed black;"/>	
466	1
606	2
612	1
803	1
940	1
<hr style="border-top: 1px dashed black;"/>	
TOTALS	496

SZIP ARE THREE LEADING DIGITS ONLY  
THIS TABLE SHOWS ACTUAL VALUES

\*NOTE: 4 DATA POINTS WERE IGNORED. 1

ALPHA DATA IN NUMERIC FIELDS: SZIP

TRANSLATION TABLE FOR FIRM

BENDER	BENDER BROTHERS	<span style="border: 1px solid black; border-radius: 50%; padding: 2px;">2</span>
ADMINI	ADMINISTRATIVE SERVICES	
CHESTE	CHESTERFIELDS	<span style="border: 1px solid black; border-radius: 50%; padding: 2px;">3</span>
DART A	DART AND GRIFFIN	
FACILI	FACILITIES MANAGEMENT	
EASY M	EASY METHOD	

\*RESPOND WITH AN OPTION NUMBER.

-->4

Figure 5.8b. This shows the concluding portion of the CROSSTAB operations where the actual values are tabulated. Note: 1) the cautionary remark concerning missing data items; 2) the translation table; and 3) the option of obtaining a cross-tabulation in terms of percentages rather than actual values.

COMPANY STAFF DISTRIBUTION BY ZIP

SZIP	MISC	BENDER	ADMINI	CHESTE	DART A	FACILI	EASY M
100	0	0	0	1.58	0	0	0
172	0	0	.93	0	0	0	0
198	0	2.32	0	0	0	0	0
200	17.35	6.97	32.71	20.63	13.33	15.62	20
204	0	2.32	0	0	0	0	0
207	33.05	32.55	25.23	28.57	42.85	28.12	28
208	18.18	18.6	10.28	19.04	18.09	9.37	16
209	7.43	9.3	5.6	14.28	5.71	3.12	4
210	1.65	0	.93	0	.95	3.12	0
217	14.87	16.27	17.75	11.11	8.57	34.37	28
220	2.47	6.97	1.66	1.58	2.85	3.12	4
221	2.47	0	0	1.58	.95	3.12	0
222	.82	0	.93	0	1.9	0	0
223	.82	2.32	.93	0	0	0	0
229	0	0	0	0	.95	0	0

803	0	0	0	0	.95	0	0
940	.82	0	0	0	0	0	0
TOTALS	100	100	100	100	100	100	100

SZIP ARE THREE LEADING DIGITS ONLY  
THIS TABLE SHOWS % OF TOTAL IN FIRM

NOTE: 4 DATA POINTS WERE IGNORED.

ALPHA DATA IN NUMERIC FIELDS: SZIP  
TRANSLATION TABLE FOR FIRM

\*RESPOND WITH AN OPTION NUMBER.

--> 0

①

\*DO YOU WANT HISTOGRAMS FOR ANY VECTOR --> yes

\*RESPOND WITH AN OPTION NUMBER

- 1 ACTUAL VALUE
- 2 % OF TOTAL IN FILE
- 3 % OF TOTAL IN SZIP

②

Figure 5.8c. These numbers show a distribution among ZIP codes by percentage of the total in each FIRM. Here the column showing total in each row has no meaning and is, therefore, not presented. Note that: 1) a zero here ends the tabulations in the present row—column orientation (transposition of rows and columns can be requested later as in fig. 5.8e); and 2) we can now get a more graphic output as shown in the next figure.

```

*****
* --> 1
* THERE ARE 7 FIRMS--MISC, BENDER BROTHERS, ADMINISTRATIVE SERVICES
* CHESTERFIELDS, DART AND GRIFFIN, FACILITIES MANAGEMENT, EASY METHOD
* ---AND TOTAL
* *SZIP VS FIRM --> total
* HISTOGRAM OF SZIP FOR TOTAL
* SZIP      %      CUM %  FREQ
* 100      .2      .2      1
* 172      .2      .4      1
* 198      .2      .6      1
* 200      19.4    20.0    96      XXXXXXXXXXXXXXXXXXXXX
* 204      .2      20.2    1
* 207      32.3    52.50   160     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
* 208      15.9    68.40   79      XXXXXXXXXXXXXXXXXXXXX
* 209      7.3     76.70   36      XXXXXXXX
* 210      1       76.70   5       X
* 217      15.7    92.40   78      XXXXXXXXXXXXXXXXXXXXX
* 220      2.8     95.20   14      XXX
* 221      1.2     96.40   6       X
* 222      .8      97.20   4       X
*
* 803      .2      99.80   1
* 940      .2      100.00  1
* *TYPE ANOTHER FIRM OR ALL FOR MORE HISTOGRAMS OF
* OPTION 1 TYPE END TO CHANGE OPTIONS --> bender brothers
* HISTOGRAM OF SZIP FOR FIRM BENDER BROTHERS
* SZIP      %      CUM %  FREQ
* 10        0      0      0
* 172      0      0      0
* 198      2.3    2.3    1      XX
* 200      7      9.3    3      XXXXXX
* 204      2.3    11.6   1      XX
* 207      32.6   44.2   14     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
* 208      18.6   62.8   8      XXXXXXXXXXXXXXXXXXXXX
* 940      0      100.00  0
* *TYPE ANOTHER FIRM OR ALL FOR MORE HISTOGRAMS OF
* OPTION 1 TYPE END TO CHANGE OPTIONS --> end
* *RESPOND WITH AN OPTION NUMBER
* --> 0
*****

```

Figure 5.8d. Now we exercise the option of obtaining frequency distributions for any or all of the columns presented earlier. Note that each of the histograms is normalized separately since they are requested separately. Had we asked for histograms for ALL the columns shown in earlier figures, we would be given an option of normalizing either uniformly over the set or individually as was done here.

\*DO YOU WANT THE ABOVE MATRIX TRANSPOSED -->yes  
 \*RESPOND WITH AN OPTION NUMBER.

--->1

\*AN HEADING --->employee distribution among zip codes  
 \*ANY FOOTNOTE --->szip are three leading digits only

EMPLOYEE DISTRIBUTION AMONG ZIP CODES

FIRM	100	172	198	200	204	207	208
MISC	0	0	0	21	0	40	22
BENDER BR	0	0	1	3	1	14	8
ADMINISTR	0	1	0	35	0	27	11
CHESTERFI	1	0	0	13	0	18	12
DART AND	0	0	0	14	0	45	19
FACILITIE	0	0	0	5	0	9	3
EASY METH	0	0	0	5	0	7	4
TOTALS	1	1	1	96	1	160	79

FIRM	209	210	217	220	221	222	223
MISC	9	2	18	3	3	1	1
BENDER BR	4	0	7	3	0	0	1
ADMINISTR	6	1	19	2	0	1	1
CHESTERFI	9	0	7	1	1	0	0
DART AND	6	1	9	3	1	2	0
FACILITIE	1	1	11	1	1	0	0
EASY METH	1	0	7	1	0	0	0
TOTALS	36	5	78	14	6	4	3

FIRM	803	940	TOTAL
MISC	0	1	121
BENDER BR	0	0	43
FACILITIE	0	0	32
EASY METH	0	0	25
TOTALS	1	1	496

Figure 5.8e. Here we see a transposition of the array shown in figure 5.8a where the rows and columns have been interchanged. Had we selected this format originally, we probably would want to transpose it to look like figure 5.8a.

```

* * * * *
* *TYPE LABELS TO BE CROSSTABBED -->? szip,firm
* *RESPOND WITH AN OPTION NUMBER.
* 1 ACTUAL VALUE
* 2 % OF TOTAL IN FILE
* 3 % OF TOTAL IN SZIP
* 4 % OF TOTAL IN FIRM
* 0 TO STOP
* --->? 2
* *ANY HEADING -->? no
* *ANY FOOTNOTE -->? crosstab after rejecting the column misc
*
* * * * FIRM * * *
* SZIP BENDER ADMINI CHESTE DART A FACILI EASY M TOTAL
* -----
* 10 0 0 .26 0 0 0 .26
* 172 0 .26 0 0 0 0 .26
* 198 .26 0 0 0 0 0 .26
* 200 .8 9.33 3.46 3.73 1.33 1.33 19.98
* 204 .26 0 0 0 0 0 .26
* 207 3.73 7.2 4.8 12 2.4 1.86 31.99
* 208 2.13 2.93 3.2 5.06 .8 1.06 15.18
* 209 1.06 1.6 2.4 1.6 .26 .26 7.18
* 210 0 .26 0 .26 .26 0 .78
* 217 1.86 5.06 1.86 2.4 2.93 1.86 15.97
* 220 .8 .53 .26 .8 .26 .26 2.91
* 221 0 0 .26 .26 .26 0 .78
* 222 0 .26 0 .53 .0 0 .79
* 223 .26 .26 0 0 0 0 .52
* 229 0 0 0 .26 0 0 .26
* 244 0 0 0 .26 0 0 .26
* 254 0 .26 0 .26 0 0 .52
* 326 0 .26 0 0 0 0 .26
* 466 0 0 .26 0 0 0 .26
* 606 .26 0 0 .26 0 0 .52
* 612 0 .26 0 0 0 0 .26
* 803 0 0 0 .26 0 0 .26
* -----
* TOTALS 11.42 28.470 16.76 27.940 8.5 6.63 99.720
* CROSSTAB AFTER REJECTING THE COLUMN MISC
* THIS TABLE SHOWS % OF TOTAL IN FILE
*
* *NOTE: 3 DATA POINTS WERE IGNORED.
* ALPHA DATA IN NUMERIC FIELDS:SZIP
* * * * *

```

Figure 5.8f. Here we see the format of the CROSSTAB output when the data presented are percentages of the total in the file. Now we get both column and row totals since they have meaning.

\* \* \* \* \*  
 \* \*RESPOND WITH AN OPTION NUMBER.\*

-->? 3

\*ANY HEADING -->? no

\*ANY FOOTNOTE -->? no

\* \* \* FIRM \* \* \*

SZIP	BENDER	ADMINI	CHESTE	DART A	FACILI	EASY M	TOTAL
100	0	0	100	0	0	0	100
172	0	100	0	0	0	0	100
198	100	0	0	0	0	0	100
200	4	46.66	17.33	18.66	6.66	6.66	99.970
204	100	0	0	0	0	0	100
207	11.66	22.5	15	37.5	7.5	5.83	99.99
208	14.03	19.29	21.05	33.33	5.26	7.01	99.970
209	14.81	22.22	33.33	22.22	3.7	3.7	99.980
210	0	33.33	0	33.33	33.33	0	99.99
217	11.66	31.66	11.66	15	18.33	11.66	99.970
220	27.27	18.18	9.09	27.27	9.09	9.09	99.990
221	0	0	33.33	33.33	33.33	0	99.99
222	0	33.33	0	66.66	0	0	99.99
223	50	50	0	0	0	0	100
229	0	0	0	100	0	0	100
244	0	0	0	100	0	0	100
606	50	0	0	50	0	0	100
612	0	100	0	0	0	0	100
803	0	0	0	100	0	0	100

THIS TABLE SHOWS % OF TOTAL IN SZIP

\*NOTE: 3 DATA POINTS WERE IGNORED.

ALPHA DATA IN NUMERIC FIELDS: SZIP

? ok

TRANSLATION TABLE FOR FIRM

BENDER BENDER BROTHERS

ADMINI ADMINISTRATIVE SERVICES

\*RESPOND WITH AN OPTION NUMBER.

-->? 0

\*DO YOU WANT HISTOGRAMS FOR ANY VECTOR -->? no

\*DO YOU WANT THE ABOVE MATRIX TRANSPOSED -->? no

Figure 5.8g. A run of the CROSSTAB module to produce a table showing percentage distribution among companies of persons whose ZIP codes begin with the three digits listed in the first column. Here column totals have no meaning and are not printed.

## 5.9 DESCRIBE

This module is designed to give a brief narrative description of the file in current use and to explain to the user the nature of the information contained in each of the data vectors in that data file. Upon entering the DESCRIBE module, the narrative description is automatically given, if one exists. Then, if the file in use is FNDEMO, and if the response to the instruction:

TYPE A LABEL(S) OR 'ALL'---> ?

is SCD, the module responds with the following

SCD, VECTOR #8, CONTAINS THE SERVICE COMPUTATION  
DATE WRITTEN AS 6 DIGITS IN THE ORDER MMDDYY

If a description is desired for the label TOA, this module will respond with:

TOA, VECTOR #10, CONTAINS A SINGLE NUMERIC  
(1-9) CODE DESCRIBING THE TYPE OF APPOINTMENT.  
\*DO YOU WISH TO SEE THE CODE---> ?

A response of 'YES' results in printout as follows:

CODES FOR DATA VECTOR TOA ARE:  
1 = COMPETITIVE CAREER  
2 = COMPETITIVE CAREER-CONDITIONAL  
3 = TAPER  
4 = TERM, INDEFINITE  
5 = COMPETITIVE TEMPORARY  
6 = EXCEPTED PERMANENT  
7 = EXCEPTED CONDITIONAL  
8 = EXCEPTED INDEFINITE  
9 = EXCEPTED TEMPORARY

A response of NO to the above question produces:

TYPE ANOTHER LABEL OR 'END'--->

The availability of this information at the terminal is a boon to the person unfamiliar with the way the information is coded in the data base. It should prove useful as a handy reminder to the data base manager as well.

The response to the request for LABEL(S) can contain more than one label. In this case, the question "Do you wish to see the codes?" is asked for each label in turn after the comment is printed. If the answer to the above requests for labels is 'ALL', this module will describe as many of the data elements as have been documented in the file including comments and codes where applicable.

If the file-description (dictionary) is not found in the main file, this module prints:

"SORRY, THIS FILE HAS NOT YET BEEN DESCRIBED"

- 01=Radio
- 02=Television
- 03=Nursing Journals
- 04=Magazines
- 05=High school teachers
- 06=High school advisors
- 07=College counselors

- 1 = I strongly agree with the statement.
- 2 = I tend to agree with the statement.
- 3 = I have no opinion or am uncertain.
- 4 = I tend to disagree with the statement.
- 5 = I strongly disagree with the statement.
- 6 = I don't care.
- 9 = Does not apply.

- 01 = East Coast
- 02 = South East
- 03 = Mid-West
- 04 = West Coast
- 05 = Alaska
- 06 = Hawaii
- 07 = Vietnam
- 08 = Korea
- 09 = Japan
- 10 = Okinawa
- 11 = Thailand
- 12 = Germany
- 13 = Italy
- 14 = Belgium
- 15 = Ethiopia
- 16 = Iran
- 17 = Canal Zone
- 99 = No preference

- 1 = Essential
- 2 = Very important
- 3 = Important
- 4 = Of little importance
- 5 = No importance whatsoever

- 80 Anatomy
- 81 Bacteriology
- 82 Biochemistry
- 83 Biology
- 84 Biophysics
- 85 Chemistry
- 86 Physics
- 87 Physiology
- 88 Mathematics
- 89 Microbiology
- 90 Statistics
- 91 Other

- 1 = First Army Area
- 2 = Third Army Area
- 3 = Fifth Army Area
- 4 = Sixth Army Area
- 5 = Military District of Washington
- 6 = Asia/Pacific
- 7 = Europe
- 8 = Middle East/Africa
- 9 = Does not apply

Figure 5.9a. Examples of typical displays of the meanings of encoded data vectors.

When this module exhausts the information on the description file without finding the desired information, it prints:

**"INFO ON XYZ IS NOT AVAILABLE"**

where XYZ is the name of the data vector in question.

The information for this module is written in dictionary records contained in the particular data base and produced by the **DICTIONARY** module discussed in section 6. The dictionary file is used also by the **SURVEY** module. Figure 5.33f shows the utility of such explicit information on otherwise implicit information in a coded file.

### **5.10 DISPLAY**

Most of the features of the **DISPLAY** module have been described in section 3.2. Here we supplement that information with a number of applications to a variety of data bases.

In figure 5.10a we show how chemical names are entered in one of the crystal data files containing both organic and inorganic compounds. At [A] we ask for the name to be displayed and find blanks in the first five records. This serves to remind us that the file does not contain the names for the organic compounds which are at the beginning of the file. Now we wish to locate where the inorganic crystals start in the file so we skip to record 500 and learn at [B] that we have not yet reached the inorganic portion. On skipping to record 700 we see at [C] that we are indeed now in the inorganic section. The remainder of the operation shows how we were able to locate record 669, which is the start of the inorganic section of this file.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >crysanor
*
* GOOD MORNING, WELCOME TO OMNIDATA
* * * NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH
*   MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? > display
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? > name
*
* REC#  NAME
* -----
* 1
* 2
* 3
* 4
* 5
*   *MORE --->? > skip to 500
*   *MORE --->? > 1
* 500
*   *MORE --->? > skip to 700
*   *MORE --->? > 1
* 700  SODIUM MANGANENE TITANTUM SILICATE PHOSPHATE*
*   *MORE --->? > back to 650
*   *MORE --->? > 1
* 650
*   *MORE --->? > skip to 675
*   *MORE --->? > 1
* 675  ARSENIC LEAD SULFIDE (4+3+9)
*   *MORE --->? > back to 660
*   *MORE --->? > 1
* 660
*   *MORE --->? > yes
* 661
* 662
* 663
* 664
* * * * *

```

Figure 5.10a. A record of the use of the DISPLAY module to locate the beginning of the inorganic section of a portion of the crystal data file. See the text for comments on the marked sections.

```

* * * * *
* 665 *
* 666 *
* 667 *
* 668 *
* 669 POTASSIUM HYDROXIDE-FLUORIDE ALUMINATE-SILICATE (1+2+4+10)*
* 670 POTASSIUM HYDROXIDE-FLUORIDE ALUMINATE-SILICATE (1+2+4+10)*
* *MORE -->? >yes *
* 671 YTTERIUM TITANIUM OXIDE (6+1+11) *
* 672 ARSENIC LEAD SULFIDE (4+3+9) *
* 673 IRON SULFATE (2+3+12) 10-HYDRATE *
* 674 CALCIUM SILICATE (1+1+3) *
* 675 ARSENIC LEAD SULFIDE (4+3+9) *
* 676 LEAD TIN ANTIMONY SULFIDE (5+3+2+14) *
* 677 *
* 678 *
* 679 ALUMINUM MAGNESIUM HYDROXIDE SILICATE (1+5+8+4+10)*
* 680 PHOSPHORUM THIO IODIDE (4+3+2) *
* *MORE -->? >end *
* *
* CPU SEC IN DISPLAY = .2102 COST = <INSTALLATION DEPENDENT*
* CPU SEC = .5196 TIME = 7:20:42 *
* *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* -->? >stop *
* *
* NORMAL EXIT BY LSD RBASIC. *
* >@fin *
* * * * *

```

Figure 5.10a (concluded).

## 5.11 DISTRIBUTE

The DISTRIBUTE module can divide a file into as many as 50 subfiles on the basis of information it finds in any one of the data vectors in the file. If a file contains a data vector called STATE giving the states of the union (as a two-letter abbreviation) and the file is distributed on STATE, this module separates out all the records belonging to the same state and stores them on separate files, one for each state encountered.

When the entire file has been distributed, this module catalogues the subfiles under names automatically generated as follows. The name of the file becomes a concatenation of 'F' for file, the label, and value of the label. For example, if the first record carried PA as the entry for STATE, the name of the first subfile would become 'FSTATEPA'. All records from the state PA would be written on this file. The next file would carry the name of the next state that is encountered as the file is read. Had the name of the state been entered as PENN or PENNSYLVANIA, the subfile for that state would have been named FSTATEPENN or FSTATEPENNSY. The truncation in the second case results from a 12 character limit established for file names. Before exiting from this module, there is printed out the list of names of the files generated and the number of logical records in each of the files. Then this module returns the user to the main Omnidata program which asks again

\*TYPE A MODULE NAME AND/OR INSTRUCTIONS --->? >

On exiting from the DISTRIBUTE module, the user has access to the same file with which he entered the module. If it is necessary to perform operations on one of the subfiles distributed in the current run or catalogued in a previous run, they can be activated in Omnidata by means of the FETCH module.

When the distribution is completed, this module generates a temporary scratch file (FFILES) available during the life of the current run which contains the names of the files that were last distributed. This scratch file supplies information to certain of the modules which allows them to perform their operations in sequence on each of the distributed files in turn. While DISTRIBUTE lists the file names in the order generated, it alphabetizes them before writing them in the scratch file (FFILES). Those modules that operate in the TANDEM mode make use of this scratch file to enable them to perform their operations on each of the files in alphabetic order.

The files resulting from the DISTRIBUTE operation are all identical in format to the files from which they originated, including labels and pointers.

An application of this module to distribute the records in a file by division (DIV) in an organization is shown in figure 5.11a. As the file contains information for DIV = 1,2,3,4,6, we get 5 files suitably named as shown at [4]. If we had typed DIV,2,3, instead of just DIV, we would

```

* * * * *
* OMNIDATA 15:22:06 28 MAR 77
*
* PLEASE ENTER ACCOUNT NUMBER --->? >XXXXX
* TYPE PASSWORD --->? >XXX
* WHICH DATA BASE DO YOU WANT --->? >fndemo
*
* GOOD AFTERNOON, WELCOME TO OMNIDATA
* * * NOTE—OMNIDATA KEEPS A RECORD OF WHO USED WHICH* * *
* MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY*
*
* FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
* TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >monitor,50,distribute ①
*
* DISTRIBUTE ON WHICH LABEL --->? >div ③
* 50 ENTRIES DISTRIBUTED TO 5 FILES 4.4272 152340
* 100 ENTRIES DISTRIBUTED TO 5 FILES 4.6032 152348
* 150 ENTRIES DISTRIBUTED TO 5 FILES 4.7836 152354
* 200 ENTRIES DISTRIBUTED TO 5 FILES 4.9564 152359
* 250 ENTRIES DISTRIBUTED TO 5 FILES 5.1346 152406
* 300 ENTRIES DISTRIBUTED TO 5 FILES 5.3064 152417
* 350 ENTRIES DISTRIBUTED TO 5 FILES 5.4726 152428
* 400 ENTRIES DISTRIBUTED TO 5 FILES 5.6362 152438
* 450 ENTRIES DISTRIBUTED TO 5 FILES 5.8174 152444
* 500 ENTRIES DISTRIBUTED TO 5 FILES 5.9934 152457
* YOU HAVE CREATED 5 FILES
* NAME # ENTRIES ②
* FDIV3 72
* FDIV2 171 ④
* FDIV6 26
* FDIV4 95
* * * * *

```

Figure 5.11a. A record of the operation of the DISTRIBUTE module on a file of 500 logical records. Note: 1) the request to monitor the operation each time 50 records are read; 2) the time in cpu seconds; 3) the elapsed clock time; and, 4) the way the subfiles are named and the number of records in each.

end up with 3 files. Records for divisions 1 and 2 would be in the first file, those for division 3 in the second file and the remainder in the third. The files would be named FDIV2, FDIV3, and FDIVREM. The instructions DIV,2,3,6 would result in the same subfiles but the name of the last file would then be FDIV6.

As the normal objective in distributing records to subfiles is to use these at a later time, the situation can easily arise that a subsequent distribute operation would result in file names that duplicate ones assigned earlier. This module, therefore, checks the list of catalogued files and appends a number 1,2,3, etc. to the file name to avoid the duplication.

In figure 5.11b we see how a bibliographic data file is distributed into 5 files, according to year of publication so that papers published before 1960 are in the first file called FYR60, those between 1961 and 1970 in the second, etc., and those later than 1976 in a file named FYRREM.

```

* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* --->? > distribute,monitor,200 *
* * * * *
* *DISTRIBUTE ON WHICH LABEL --->? > YR,60,70,75,76 *
* 200 ENTRIES DISTRIBUTED TO 6 FILES 18.5554 141300 *
* 400 ENTRIES DISTRIBUTED TO 6 FILES 19.1478 141322 *
* 600 ENTRIES DISTRIBUTED TO 6 FILES 19.7576 141355 *
* 800 ENTRIES DISTRIBUTED TO 6 FILES 20.377 141428 *
* YOU HAVE CREATED 5 FILES *
* NAME # ENTRIES *
* FYR60 28 *
* FYR70 245 *
* FYR75 498 *
* FYR76 97 *
* FYRREM 8 *
* * * * *

```

Figure 5.11b. Here we see how the DISTRIBUTE module allows for creation of subfiles containing specific desired groupings of records rather than a single record type. See the text for further discussion of this figure.

If one wished to distribute a data file into subfiles containing states in the New England region, the mid-Atlantic region, etc., it can be accomplished by using two modules. AGGREGATE can be used first, as shown in figure 5.02d, to generate a new data vector called *region*. Next, if the DISTRIBUTE module is instructed to operate on the *region* vector, the subfiles would contain the desired groupings of states. A similar operation on a data file containing names of countries would allow for aggregating and then distributing to subfiles by continent.

On the other hand, if we wished to DISTRIBUTE records to subfiles containing specified numbers of countries per record, this module will do so, provided we are satisfied to have them appear in groups in alphabetic order. Thus, if we respond to the question

\*DISTRIBUTE ON WHICH LABEL?

by typing

COUNTRY,BOTSWANA,C,GAMBIA,KUWAIT,SWA,ZANZIBAR,

the result will be 7 files. The first would contain records for countries whose names fall between Afghanistan and Botswana. The second file would contain words for the countries Brazil to Czechoslovakia, etc. The seventh file will contain countries beyond ZANZIBAR. If there are none, the file is empty.

It should be noted that in none of the above cited applications is there a requirement that the original file be in sorted order. Nor will the resulting subfiles necessarily be in sorted order. The records will be written in the order in which they appear in the original file.

This feature of building subfiles containing clusters of records on the basis of alphabetic (or numeric) order of a particular data vector is useful when it is necessary to SORT a large file. Since shorter files can be sorted much more efficiently than larger ones, the sorted subfiles can then be combined via the STACK module to produce a sorted file.

\* \* \* N O T E S \* \* \*

## 5.12 ENCODE

Effective analysis of information in detailed data files requires facilities for encoding the information. Preparation of summaries, frequency distributions, and cross-tabulations are facilitated by being able to replace the wide variety of numeric values by relatively few classes: age groups, salary groups, or any other grouping that will make the display and analysis more manageable. The ENCODE module is useful for this purpose. It works in two modes which are selected by typing either A(UGMENT) or R(EPLACE) in response to the first input request. In the former case, the module adds a *new* data vector to the file. In the latter case, the original data items are replaced by the appropriate code.

If we wish to encode the PAY vector in the augmenting mode, the module asks for the name of the new vector. The advantage of using the augment mode is that the original data are still available for other purposes or for encoding into narrower classes. Obviously, encoding again to broader classes can be achieved by recoding the encoded vector, but with less control over the class intervals. After the mode has been selected, this module asks for an input

INPUT LABEL TO BE ENCODED AND INTERVALS

--> ?

If the response is

PAY, 5000, 10000, 12000, 14000, 16000, 18000, 20000  
25000, 30000

the actual salary figure would be replaced by a number indicating the group to which that figure belongs. If the salary in a particular record was 7530, it would fall into group 2. Entries between 10,000 and 11,999 would fall into group 3, etc. Salaries above and below the designated range are also counted and reported. Thus, salaries below 5000 would be put into group 1, while those above 30,000 would be put into group 10. See figure 5.12a for the results of this encoding operation and the dialogue required to achieve them.

In the above example, we specified the precise boundaries of the class intervals we wished to establish for encoding. This operation becomes tedious when the precise boundaries are really not crucial to the subsequent analysis. In many instances classification into 10 classes would be sufficient. Such encoding is indeed available and can be exercised by responding with a label and the number 10. In order to achieve this, it is necessary for the module to know what is the minimum and maximum data entry in the particular vector. The OMNIDATA file format allows for storing such information in a summary table. If a summary table exists in the file in question, ENCODE reads it and uses it to set up class intervals.

If users prefer to ignore these actual extreme data points, they may supply their own upper and lower bounds by typing, for example, PAY, 10, 6000-36000. This instruction would generate 10 uniformly spaced intervals between 6000 and 36,000 and enter into a new vector a number for each record indicating into which interval the pay value falls. Since there are likely to be pay values falling outside of these limits, the intervals are numbered 2 to 11 instead of 1 to 10. This module performs similarly for intervals other than 10, except that there is a limit of 50 intervals.

If the file to be encoded has not been summarized previously, this module requests the user to supply lower and upper bounds. In figure 5.12a we see how this module responds in such a case. Here we did not specify the number of intervals, so the ENCODE module assumed that 10 intervals would be satisfactory and asked for the lower and upper bounds since it did not find a file summary. When a large number of uniform intervals are required, the information can be supplied more economically by indicating the lower bound, the interval size, and the upper bound. Thus, the input AGE, 10(10)50 will set up 6 class intervals as follows: <10, 10-20, 20-30, 30-40, 40-50, 50>.

After each of the designated vectors have been encoded, the user is offered the opportunity to obtain histograms. At this stage it is possible to have each of the class intervals identified either by their mid-range or by the class interval. Figures 5.12a et seq. show the variety of input and output options available in this module.

\* \* \* N O T E S \* \* \*

```

* * * * *
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   --->? >encode
*   *WHICH MODE? TYPE R(EPLACE) OR A(UGMENT) --->? >augment
*   *INPUT LABEL TO BE ENCODED AND INTERVALS
*   --->? >pay,5000,10000,12000,14000,18000,20000,25000,30000
*   --->? >age
*   *TYPE LOWER,UPPER BOUNDS TO BE DIVIDED INTO 10 CLASSES
*   --->? >20,70
*   --->? >yos,5(5)40
*   --->? >backup,1
*   --->? >yos,5(5)45
*   --->? >end
*   *TYPE THE NEW LABEL FOR THE ENCODED PAY --->? >cpay
*   *TYPE THE NEW LABEL FOR THE ENCODED AGE --->? >dob
*   DOB IS ALREADY A LABEL-PLEASE SELECT ANOTHER WORD
*   *TYPE THE NEW LABEL FOR THE ENCODED AGE --->? >cage
*   *TYPE THE NEW LABEL FOR THE ENCODED YOS --->? >ayos
*
*   FREQUENCY DISTRIBUTION OF PAY
*   74 : 96 43 37 60 26 77 40 : 47
*
*   FREQUENCY DISTRIBUTION OF YOS
*   120 : 100 71 67 62 36 34 3 1 : 6
*
*   *INPUT LABELS FOR HISTOGRAMS DESIRED --->? >all
*
*   HISTOGRAM OF PAY DISTRIBUTION
*   CLASS INT.      VALUE
*   1      <5000      74 : XXXXXXXXXXXXXXXXXXXXXXXX
*   2      5000-10000 96 : XXXXXXXXXXXXXXXXXXXXXXXX
*   3      >10000-12000 43 : XXXXXXXXXXXXXXXX
*   4      >12000-14000 37 : XXXXXXXXXXXXXXXX
*   5      >14000-18000 60 : XXXXXXXXXXXXXXXXXXXXXXXX
*   6      >18000-20000 26 : XXXXXXXXX
*   7      >20000-25000 77 : XXXXXXXXXXXXXXXXXXXXXXXX
*   8      >25000-30000 40 : XXXXXXXXXXXXXXXX
*   9      >30000      47 : XXXXXXXXXXXXXXXX
*
* * * * *

```

Figure 5.12a. Here we have a record of the use of the ENCODE module to augment a data file with three new vectors containing classification into designated class intervals of the data on AGE, PAY and YOS (years of service). *Note:* 1) alternate ways of specifying the class intervals; 2) the use of the BACKUP command to correct a previous entry; 3) the way in which names are assigned to the new data vectors; 4) the frequency distributions for each of the encoded data vectors; and 5) the appearance of the histograms. As many as 10 data vectors can be encoded by this module in one pass through the file.

```

* * * * *
* HISTOGRAM OF AGE DISTRIBUTION
* CLASS INT.      VALUE
* 1      <20      25 : XXXXXXXXXXXX
* 2      >20-25   48 : XXXXXXXXXXXXXXXXXXXXX
* 3      >25-30   42 : XXXXXXXXXXXXXXXXXXXXX
* 4      >30-35   49 : XXXXXXXXXXXXXXXXXXXXX
* 5      >35-40   59 : XXXXXXXXXXXXXXXXXXXXX
* 6      >40-45   73 : XXXXXXXXXXXXXXXXXXXXX
* 7      >45-50   75 : XXXXXXXXXXXXXXXXXXXXX
* 8      >50-55   63 : XXXXXXXXXXXXXXXXXXXXX
* 9      >55-60   45 : XXXXXXXXXXXXXXXXXXXXX
* 10     >60-65   15 : XXXXXX
* 11     >65-70   6  : XX
* 12     >70     0  :
*
* HISTOGRAM OF YOS DISTRIBUTION
* CLASS INT.      VALUE
* 1      <5       120 : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
* 2      5-10    100 : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
* 3      >10-15  71  : XXXXXXXXXXXXXXXXXXXXXXXX
* 4      >15-20  67  : XXXXXXXXXXXXXXXXXXXXXXXX
* 5      >20-25  62  : XXXXXXXXXXXXXXXXXXXXXXXX
* 6      >25-30  36  : XXXXXXXXXXXXX
* 7      >30-35  34  : XXXXXXXXXXXXX
* 8      >35-40   3  : X
* 9      >40-45   1  :
* 10     >45     0  : XX
*
* *MORE --->? >yes
* *INPUT LABELS FOR HISTOGRAMS DESIRED --->? >age,mid
*
* HISTOGRAM OF AGE DISTRIBUTION
* MIDRANGE      VALUE
* 1      < 20    25 : XXXXXXXXXXXX
* 2      22.5   48 : XXXXXXXXXXXXXXXXXXXXX
* 3      27.5   42 : XXXXXXXXXXXXXXXXXXXXX
* 4      32.5   49 : XXXXXXXXXXXXXXXXXXXXX
* 5      37.5   59 : XXXXXXXXXXXXXXXXXXXXX
* 6      42.5   73 : XXXXXXXXXXXXXXXXXXXXX
* 7      47.5   75 : XXXXXXXXXXXXXXXXXXXXX
* 8      52.5   63 : XXXXXXXXXXXXXXXXXXXXX
* 9      57.5   45 : XXXXXXXXXXXXXXXXXXXXX
* 10     62.5   15 : XXXXXX
* 11     67.5   6  : XX
* 12     > 70   0  :
*
* *MORE --->? >no
* * * * *

```

Figure 5.12b. A continuation of the output from the run started in the previous figure showing: 1) normal output in terms of class intervals, and 2) output in terms of the mid-points of the interval. The next figure shows a cross-tabulation of two of the encoded data vectors.



```

* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >display
*
* *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >pay,cpay,age,cage,yos,cyos
*
* PAY      CPAY      AGE      CAGE      YOS      CYOS
* -----
* 08027    2          37      5          6        2
* 32280    9          59      9          33       7
* 15860    5          52      8          25       5
* 21671    7          37      5          11       3
* 26938    8          52      8          31       7
* 19462    6          46      7          12       3
* 13162    4          52      8          16       4
* 30486    9          49      7          24       5
* 31383    9          45      6          11       3
* 34323    9          49      7          14       3
*
* *MORE --->? >30
* 20357    7          32      4          5        1
* 08943    2          55      8          29       6
* 18350    6          36      5          8        2
* 22328    7          53      8          10       2
* 22985    7          40      5          13       3
*
*
* 13627    4          53      8          5        1
* 10471    3          52      8          10       2
* 13996    4          37      5          5        1
* 35363    9          62     10         32       7
* 21671    7          60      9          19       4
* 21671    7          34      4          8        2
* 36000    9          52      8          23       5
*
* *MORE --->? >no
* *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >end
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* PROGRAM STOPPED.
* * * * *

```

Figure 5.12d. A display of the data items PAY, AGE, and YOS and their equivalent class designations (codes) CPAY, CAGE, and CYOS produced by the instructions shown in figure 5.12a.

### 5.13 EXTRACT

The EXTRACT module allows the user to specify positions in a given data vector which are to be extracted to form a new vector of information. This new vector may be defined to be either string data or numeric data. Its primary application is in elevating information contained in one of the miscellaneous data vectors generated in the DEFINE module to a major Omnidata data vector. It gives the user the added capability of duplicating a vector in order to redefine it as either numeric or alphabetic data. It is sometimes convenient to be able to treat a numeric data vector either as a number or a string of characters.

If in response to the instruction:

**\*TYPE NEW LABEL, OLD LABEL, AND POSITIONS --->**

the user responds with ZIP3, ZIP, 1,3, this module extracts the first three digits of the ZIP code field and enters it as a new data vector called ZIP3. In the next two exchanges with this module the user designates whether the new item is to be treated as numeric or alphabetic and whether the data is restricted or not. Figure 5.13a shows a typical use of this feature.

There are two other ways to specify the location of the data items to be extracted and relabeled. One of these is in relation to its position in the logical record of the file. The other is in relation to the location of the data item in each of the sectors on the disc into which the original logical record is read. As an example, it is informative to consider an item of information called PLANT located in positions 278-282 of the original file and hence in sector 2 on the disc where it is located in positions 110-114. Now, if we wished to extract from these five characters the leading digit which represents the REGION, there are the following alternatives:

REGION, PLANT, 1, 1 or REGION, LOGICAL, 1, 278, 278

or REGION, SECTOR, 2, 110, 110.

Each of these gives the same results.

EXTRACT operates in two modes. According to the response to the first input request, the new vectors are generated temporarily for the run or are added to the file permanently.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >fcarpool
* FILE FCARPOOL CONTAINS 36 DATA ITEMS FOR 500 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >no cost,extract
*   *TYPE MODE - T(EMP) OR P(ERM) --->? >p
*   *TYPE NEW LABEL, OLD LABEL, AND POSITIONS --->? >zip1,zip,1,1
*   *TYPE S(TRING) OR N(UMBER) --->? >number
*   *TYPE R(ESTRICTED) OR N(ORMAL) --->? >normal
*   *TYPE NEW LABEL, OLD LABEL, AND POSITIONS --->? >zip2,zip,1,2
*   *TYPE S(TRING) OR N(UMBER) --->? >number
*   *TYPE R(ESTRICTED) OR N(ORMAL) --->? >normal
*   *TYPE NEW LABEL, OLD LABEL, AND POSITIONS --->? >zip3,zip,1,3
*   *TYPE S(TRING) OR N(UMBER) --->? >number
*   *TYPE R(ESTRICTED) OR N(ORMAL) --->? >normal
*   *TYPE NEW LABEL, OLD LABEL, AND POSITIONS --->? >y-crd,y2-crd,1,3
*   *TYPE S(TRING) OR N(UMBER) --->? >number
*   *TYPE R(ESTRICTED) OR N(ORMAL) --->? >normal
*   *TYPE NEW LABEL, OLD LABEL, AND POSITIONS --->? >end
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >save
*
*   *TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->? >fcarpool
* FCARPOOL IS CATALOGUED— 64 TRACKS 500 RECORDS 9.7778 *
*   *TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->? >end*
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* PROGRAM STOPPED.
* TIME : 11.052
* * * * *

```

Figure 5.13a. Instructions for generating five new data vectors by extracting certain digits from existing data vectors in an experimental carpool file. The new vectors X-CRD and Y-CRD are shown in figure 5.7a together with the original data vectors from which they were extracted.



## 5.15 FIT

This module fits a least-squares polynomial of degree  $m$  ( $m < 10$ ) of the form

$$Y = A_0 + A_1X + A_2X^2 + \dots A_mX^m$$

where  $X$  and  $Y$  are two numeric data vectors. This is accomplished by interfacing with the OMNITAB (1978) system, which has one of the more accurate algorithms for least-squares calculations. As our first example, we show the result of fitting a fifth degree polynomial to 17 data points representing the thermal conductivity of a copper-nickel alloy as a function of temperature.

The results of this operation are presented on four pages (see figs. 5.15a et seq.) and contain the following:

On the first page we have a tabulation of:

- a) the independent variable (TEMP),
- b) the response (the thermal conductivity for an alloy containing 95% nickel),
- c) the predicted response computed from the fitted polynomial,
- d) the standard deviation of each predicted response,
- e) the residuals (the difference between the observed and predicted values),
- f) the standardized residuals, and
- g) the weights used in the fitting operation.

It should be noted that the weights in the last column of fig. 5.15a are unity for the first 17 rows devoted to the actual data fitted. The remainder of the rows carry zero weights so the data in rows 18-37 of the second and third columns did not enter into the least-squares calculations. Their presence in the printout is for the purpose of computing a table at uniform or at specific values of the temperature. How this is accomplished is discussed later.

On the second page (fig. 5.15b) we have four plots showing:

- a) the standardized residuals versus row number,
- b) the standardized residuals versus predicted response,
- c) the standardized residuals versus the temperature, and
- d) a normal probability plot of the standardized residuals.

On the third page (fig. 5.15c), we have two types of information of interest to the statistical analyst:

- a) the variance-covariance matrix to assess the collinearity of the data vectors fitted, and
- b) the analysis of variance from which it is possible to assess the adequacy of the model (in this case a fifth degree polynomial).

LEAST SQUARES FIT OF RESPONSE, 95P NI,  
 AS A POLYNOMIAL OF DEGREE 5. INDEPENDENT VARIABLE IS  
 TEMP B  
 USING 17 NON-ZERO WEIGHTS AND 20 ZERO WEIGHTS IN COLUMN

ROW	INDEP. VAR. TEMP	RESPONSE 95P NI	PREDICTED RESPONSE	STD. DEV. OF PRED. RESPONSE	RESIDUALS	STD. RES.	WEIGHTS
1	4.000000	.02790000	.028039233	.00063935987	-.00013923313	.19	1.000
2	6.000000	.04380000	.043823853	.00050137027	-.000023853140	.03	1.000
3	8.000000	.06040000	.060368444	.00041760064	.000011555576	.01	1.000
4	10.000000	.077799999	.077616852	.00038342955	.00018314769	.21	1.000
5	15.000000	.12300000	.12288396	.00041391867	.00011604439	.13	1.000
6	20.000000	.17100000	.17006704	.00045689687	.00093295451	1.10	1.000
7	25.000000	.21700000	.21782448	.00046365076	-.00082447828	-.98	1.000
8	30.000000	.26400000	.26502100	.00044825258	-.0010210083	-1.20	1.000
9	40.000000	.35400000	.35414557	.00043719235	-.00014557475	-.17	1.000
10	50.000000	.43300000	.43200022	.00048487566	.00099977499	1.20	1.000
11	60.000000	.49700000	.49562753	.00051137026	.0013724671	1.68	1.000
12	70.000000	.54300000	.54414366	.00047987510	-.0011436687	-1.37	1.000
13	80.000000	.57999999	.57832869	.00045184083	-.00032870334	-.39	1.000
14	90.000000	.59900000	.60021711	.00053802526	-.0012171189	-1.53	1.000
15	100.00000	.61399999	.61268825	.00072143789	.0013117458	2.06	1.000
16	150.00000	.64200000	.64209272	.00096081275	-.000092722589	-1.53	1.000
17	200.00000	.63699999	.63699133	.00096249756	.0000086663880	1.33	1.000
18	5.000000	0.	.035826542	.00056381597	-.035826542	.00	0.
19	10.000000	0.	.077616850	.00038342955	-.077616850	.00	0.
20	15.000000	0.	.12288396	.00041391867	-.12288396	.00	0.
21	20.000000	0.	.17006704	.00045689687	-.17006704	.00	0.
22	25.000000	0.	.21782447	.00046365076	-.21782447	.00	0.
23	30.000000	0.	.26502100	.00044825258	-.26502100	.00	0.
24	35.000000	0.	.31071499	.00043417541	-.31071499	.00	0.
25	40.000000	0.	.35414557	.00043719235	-.35414557	.00	0.
26	45.000000	0.	.39471987	.00045765652	-.39471987	.00	0.
27	50.000000	0.	.43200022	.00048487566	-.43200022	.00	0.
28	60.000000	0.	.49562753	.00051137026	-.49562753	.00	0.
29	70.000000	0.	.54414367	.00047987510	-.54414367	.00	0.
30	80.000000	0.	.57832870	.00045184083	-.57832870	.00	0.
31	90.000000	0.	.60021712	.00053802526	-.60021712	.00	0.
32	100.00000	0.	.61268824	.00072143789	-.61268824	.00	0.
33	120.00000	0.	.62266260	.00097689735	-.62266260	.00	0.
34	140.00000	0.	.63261919	.00087202748	-.63261919	.00	0.
35	160.00000	0.	.65423025	.0013123670	-.65423025	.00	0.
36	180.00000	0.	.67336661	.0019178259	-.67336661	.00	0.
37	200.00000	0.	.63699133	.00096249756	-.63699133	.00	0.

Figure 5.15a. This is the first of four pages produced by the FIT module in the process of fitting a fifth degree polynomial to 17 data pairs at [A] and [B]. See the text for an explanation of the numbers in rows 18-37.

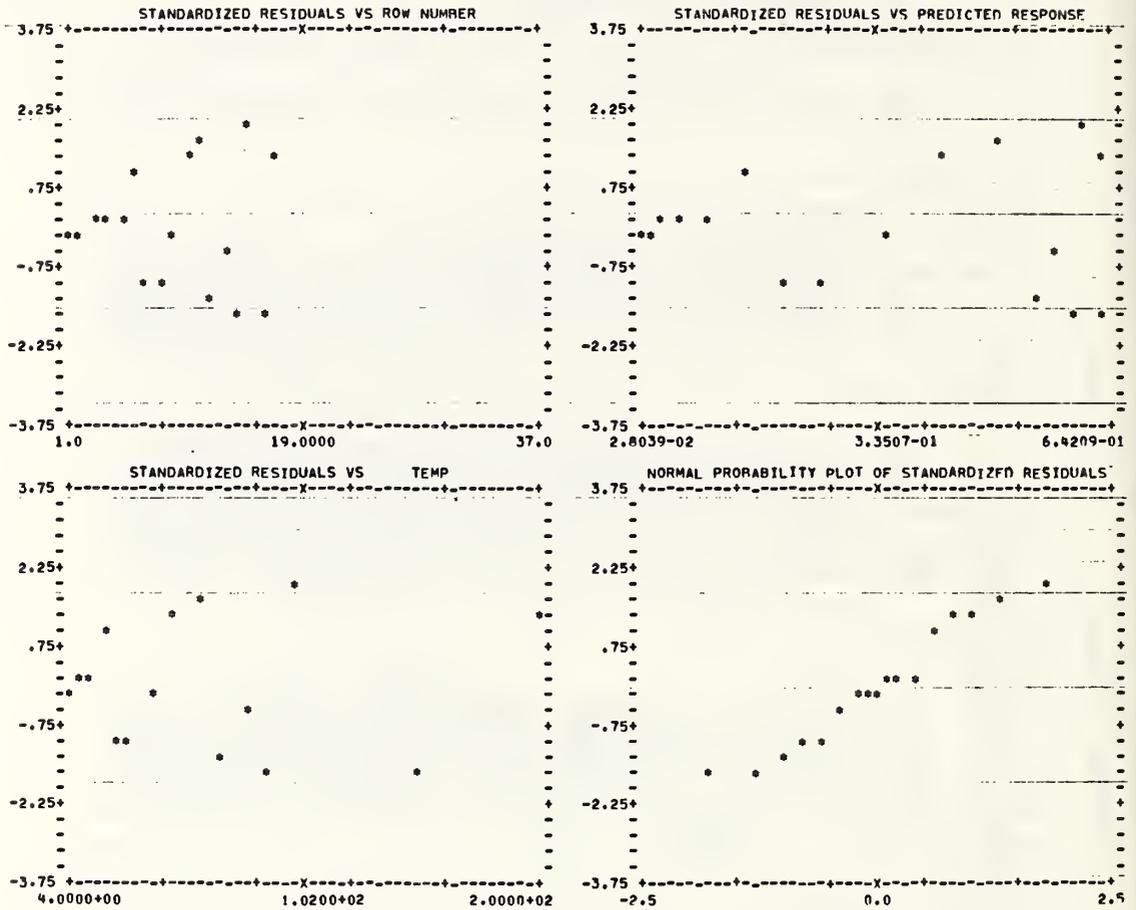


Figure 5.15b. This second page provides a variety of graphical presentations of the residuals (observed values—calculated values).

LEAST SQUARES FIT OF RESPONSE, 9SP NI,  
 AS A POLYNOMIAL OF DEGREE 5. INDEPENDENT VARIABLE IS TEMP  
 USING 17 NON-ZERO WEIGHTS AND 20 ZERO WEIGHTS IN COLUMN 8

VARIANCE-COVARIANCE MATRIX OF THE ESTIMATED COEFFICIENTS

TERM	0	1	2	3	4	5
0	1.1412259-06					
1	-1.4104146-07	2.1053395-08				
2	4.8040295-09	-7.7159473-10	2.9760318-11			
3	-6.5035199-11	1.0840526-11	-4.3113842-13	6.3713133-15		
4	3.7276496-13	-6.3457024-14	2.5717229-15	-3.8496735-17	2.3465547-19	
5	-7.5388172-16	1.3000756-16	-5.3331749-18	8.0525568-20	-4.9386764-22	1.04440629-24

ANALYSIS OF VARIANCE  
 -DEPENDENT ON ORDER INDEPENDENT VARIABLES ARE ENTERED, UNLESS VECTORS ARE ORTHOGONAL-

TERM	SS=RED. DUE TO COEF.	CUM. RESIDUAL MS	D.F.	F (COEF=0)	P (F)	F (COEFS=0)	P (F)
0	2.0351028	.055057940	16	2196681.844	.000	524590.000	.000
1	.65793780	.014865949	15	710175.461	.000	190171.631	.000
2	.21314077	.00070346211	14	230063.301	.000	60170.674	.000
3	.0078671681	.00015240781	13	8491.790	.000	3539.800	.000
4	.00093743381	.000086988974	12	1011.862	.000	1063.804	.000
5	.0010336768	.0000009264400	11	1115.747	.000	1115.747	.000
RESIDUAL	.000010190884		11				
TOTAL	2.9160298		17				

Figure 5.15c. Here we have information which helps the statistical analyst determine the collinearity of the data vectors and the appropriateness of the model (the fifth degree polynomial in this case).

On the fourth page (fig. 5.15d) we have the coefficients and related data as follows;

- a) for the fifth degree polynomial requested, and
- b) for a polynomial one degree lower (a fourth degree).

Least-squares fitting of data is usually performed at nonuniformly spaced values of the independent variable (X) with the objective of producing a smooth table of Y's at uniformly spaced values of X. Normally the coefficients from the fitting operation are used to produce such a table in a subsequent calculation. This module provides for the computation of a "smooth" table as a by-product of the least-squares fitting operation.

To produce a computed table at the desired temperatures, it is only necessary to augment the X and Y vectors and the vector of weights as shown in fig. 5.15a. There we see that lines 18 through 37 contain the list of temperatures, while the response column has been set to zero. More importantly, since the weights for rows 18-37 have been set to zero, these numbers are omitted from the least-squares fit. The module does, however, compute a predicted response for each of the 37 rows of the independent variable (TEM). Those values are shown in the printout at [C] in figure 5.15a.

The OMNITAB system has a built-in worksheet that can accommodate 12,500 numbers. It is normally dimensioned to consist of 201 rows by 62 columns. If the records in the Omnidata file exceed 201, this module and others which interface with OMNITAB adjust the dimensions of the worksheet accordingly. In figure 5.15e we have the dialogue which produced the earlier figures. The text in the response at [A] is placed on the OMNITAB card (instruction 4) so that it can be carried on each page of the OMNITAB output. At [B] see how the module is instructed to transfer two data vectors TEMP and KAT95N to OMNITAB.

The augmentation of the data vectors to produce a table of values at rounded temperatures shown in figure 5.15a is achieved as follows. At [C] we generate the desired list of temperatures at which we wish the module to compute the thermal conductivity from the fitted coefficients. The instruction at [D] provides a column of 1's to be used as weights for the 17 measured points. At [E] we use the DEMOTE instruction to place the 20 rounded temperatures underneath the 17 measured ones in column 1. The DEMOTE operation not only moves the data as instructed, but also alerts OMNITAB to operate on 37 rows instead of 17. Note that as we produced 17 unit weights in column 4 before using the DEMOTE instruction, the values in rows 18-37 of that column are still zero as can be seen in the last column of figure 5.15a.

The positive response at [F] alerts this module to make provision in the OMNITAB POLYFIT instruction (via the last two numbers in line 12) to store the residuals and coefficients in the worksheet as indicated at [G] for subsequent plotting as in line 13.

LEAST SQUARES FIT OF RESPONSE, 95P NI,  
 AS A POLYNOMIAL OF DEGREE 5. INDEPENDENT VARIABLE IS TEMP  
 USING 17 NON-ZERO WEIGHTS AND 20 ZERO WEIGHTS IN COLUMN 8

ESTIMATES FROM LEAST SQUARES FIT

TERM	COEFFICIENT	S.D. OF COEFF.	RATIO	ACCURACY*	COEFFICIENT	S.D. OF COEFF.	RATIO
0	-.00069414452	.0010682818	-.65	6.00	-.025336795	.0074862418	-3.38
1	.0066455455	.00014509788	45.80	7.36	.010895537	.00067585359	16.12
2	.00014571494	.0000054553018	26.71	7.06	-.000028628373	.000015376067	-1.86
3	-.0000028842557	.000000079820508	-36.13	7.05	-.00000025184730	.0000012280206	-2.05
4	.00000001712420	4.8441250-10	35.45	6.99	.000000010277177	3.1306580-10	3.28
5	-3.4130773-11	1.0217940-12	-33.40	7.05			

RESIDUAL STANDARD DEVIATION = .00096251962  
 BASED ON DEGREES OF FREEDOM 17- 6 = 11

.0093267879  
 17- 5 = 12

\* THE NUMBER OF CORRECTLY COMPUTED DIGITS IN EACH COEFFICIENT USUALLY DIFFERS BY LESS THAN 1 FROM THE NUMBER GIVEN HERE

Figure 5.15d. Here we have, finally, the coefficients and their related accuracy for both the fit requested and one of lower degree.

The PRINT response at [H] causes all of the results to be printed out via the POLYFIT instruction. A response of 'STORE' would cause the OMNITAB command in line 12 to start with SPOLYFIT, which carries out the least squares fitting, stores the coefficients and residuals but does not produce the printed output shown in figures 5.15a through 5.15d. The feature to suppress the printout is useful when we want to perform a series of systematic fits and wish only to compare the final coefficients or residuals. It is important to note that the STORE command refers to storing results in the OMNITAB worksheet and not on a computer file.

As we choose to perform only one fit, the negative response at [I] produces a listing of the OMNITAB commands which this module has written in response to the above instructions. Finally, a negative response at [J] results in the printout at [K] which is the normal response when the XBASIC compiler is released and the run stream starting at line 1 (@ASG,A FITTESTD) is initiated. At [L], we see the first line of the output from OMNITAB which has taken over from Omnidata. This line appears on each page of the output.

Figure 5.15f shows how the coefficients and their associated accuracies are presented. The graphical analyses of the residuals are presented on two pages, one of which is shown in figure 5.15g.

\* \* \* N O T E S \* \* \*

```

* * * * *
* *TYPE NAME FOR OMNITAB FILE ---? >fittest
* *TYPE P(ERMANENT) OR T(EMPORARY) FOR FILE ---? >t (A)
* *TYPE A TITLE FOR THIS RUN OR N(ONE)
* ? >copper nickel system (wt is weight percent of nickel)
(B) *TYPE LABELS TO BE TRANSFERRED TO OMNITAB FILE
    (SEQ IS A LEGITIMATE ENTRY) ---
* ? >temp,kat95n
* *YOUR WORKSHEET IS DIMENSIONED FOR 201 ROWS BY 62 COLUMNS
* *THE DATA ARE STORED IN THE OMNITAB WORKSHEET AS FOLLOWS:
* COLUMN LABEL
* 1 TEMP
* 2 KAT95N
*
* *INPUT ANY OMNITAB COMMANDS YOU WANT PERFORMED (C)
* BEFORE FIT - TYPE END TO END -
* ? >generate 5.(5.)50.(10.)100.(20.)200. in 3 (D)
* ? >add 1. to col 4 store in 4
* ? >demote by 17 rows col 3 into col 1 (E)
* ? >end
* *TYPE LABEL OR COL NUMBER OF DEPENDENT Y VARIABLE ---? >kat95n
* *TYPE LABEL OR COL NUMBER TO BE USED AS WEIGHTS, OR N(ONE) ---?4
* *TYPE POLYNOMIAL DEGREE ---? >5
* *TYPE LABEL OR COL NUMBER FOR INDEPENDENT X VARIABLE ---? >temp
* *DO YOU WANT RESIDUALS PLOTTED ---? >yes (F)
* COEFFICIENTS ARE IN COLUMN 62 (G)
* RESIDUALS ARE IN COLUMN 61
* *TYPE P(RINT) OR S(TORE) FOR RESULTS ---? >print (H)
* *DO YOU WANT ANY MORE FITS ---? >no (I)
* FOLLOWING IS A LIST OF YOUR OMNITAB FIT INSTRUCTIONS:
* 1 @ASG,A FITTESTD
* 2 @USE 7.,FITTESTD
* 3 @NBS*OMNITAB
* 4 OMNITAB-copper nickel system (wt is weight percent of nickel)
* 5 DIMENSION THE WORKSHEET TO HAVE 201 ROWS AND 62 COLUMNS
* 6 FORMAT A (2F12.0)
* 7 READ TAPE A A INTO COLUMNS 1*** 2
* 8 LABEL TEMP, KAT95N
* 9 GENERATE 5.(5.)50.(10.)100.(20.)200.IN 3
* 10 ADD 1. TO COL 4 STORE IN 4
* 11 DEMOTE BY 17 ROWS COL 3 INTO COL 1
* 12 POLYFIT Y IN COL 2, WTS 4, DEG 5, X IN 1 PUT IN 62 AND 61
* 13 PAGE PLOT COLS 61 VS 1
* 14 STOP (J)
* *DO YOU WISH TO CHANGE ANYTHING ---? >no
* TIME: 6.940
(K) LSD XBASIC 12:38:48 2NOV77 (L)
* READY
*
* OMNITAB-COPPER NICKEL SYSTEM (WT IS WEIGHT PERCENT OF NICKEL)
* * * * *

```

Figure 5.15e. Here we see the dialogue between the FIT module to produce a least-squares fit for the thermal conductivity of a nickel alloy from 17 measured values shown in figure 5.15a. See the text for a discussion of the marked items and the PLOT module for facilities for editing (deleting, correcting, or inserting) OMNITAB commands.

OMNITAB - COPPER NICKEL SYSTEM ( WT IS WEIGHT PERCENT OF NICKEL.

LEAST SQUARES FIT OF RESPONSE, 95P NI,  
AS A POLYNOMIAL OF DEGREE 5. INDEPENDENT VARIABLE IS TEMP  
USING 17 NON-ZERO WEIGHTS AND 20 ZERO WEIGHTS IN COLUMN A

ESTIMATES FROM LEAST SQUARES FIT

TERM	COEFFICIENT	S.D. OF COEFF.	RATIO	ACCURACY*
0	-.00069414452	.0010682818	-.65	6.00
1	.0066455455	.00014509788	45.80	7.36
2	.00014571494	.0000054553018	26.71	7.06
3	-.0000028842557	.000000079820508	-36.13	7.05
4	.000000017172420	4.8441250-10	35.45	6.99
5	-3.4130773-11	1.0217940-12	-33.40	7.05

RESIDUAL STANDARD DEVIATION = .00096251962  
BASED ON DEGREES OF FREEDOM 17- 6 = 11

\* THE NUMBER OF CORRECTLY COMPUTED DIGITS IN EACH COEFFICIENT  
USUALLY DIFFERS BY LESS THAN 1 FROM THE NUMBER GIVEN HERE.

FIT OMITTING LAST TERM

TERM	COEFFICIENT	S.D. OF COEFF.	RATIO
0	-.025338795	.0074862418	-3.38
1	.010895537	.00067585359	16.12
2	-.000028628373	.000015376067	-1.86
3	-.00000025184730	.00000012280206	-2.05
4	.0000000010277177	3.1306580-10	3.28

RESIDUAL STANDARD DEVIATION = .0093267879  
BASED ON DEGREES OF FREEDOM 17- 5 = 12

Figure 5.15f. Here we see how this module formats the output when it is directed to a narrow terminal.

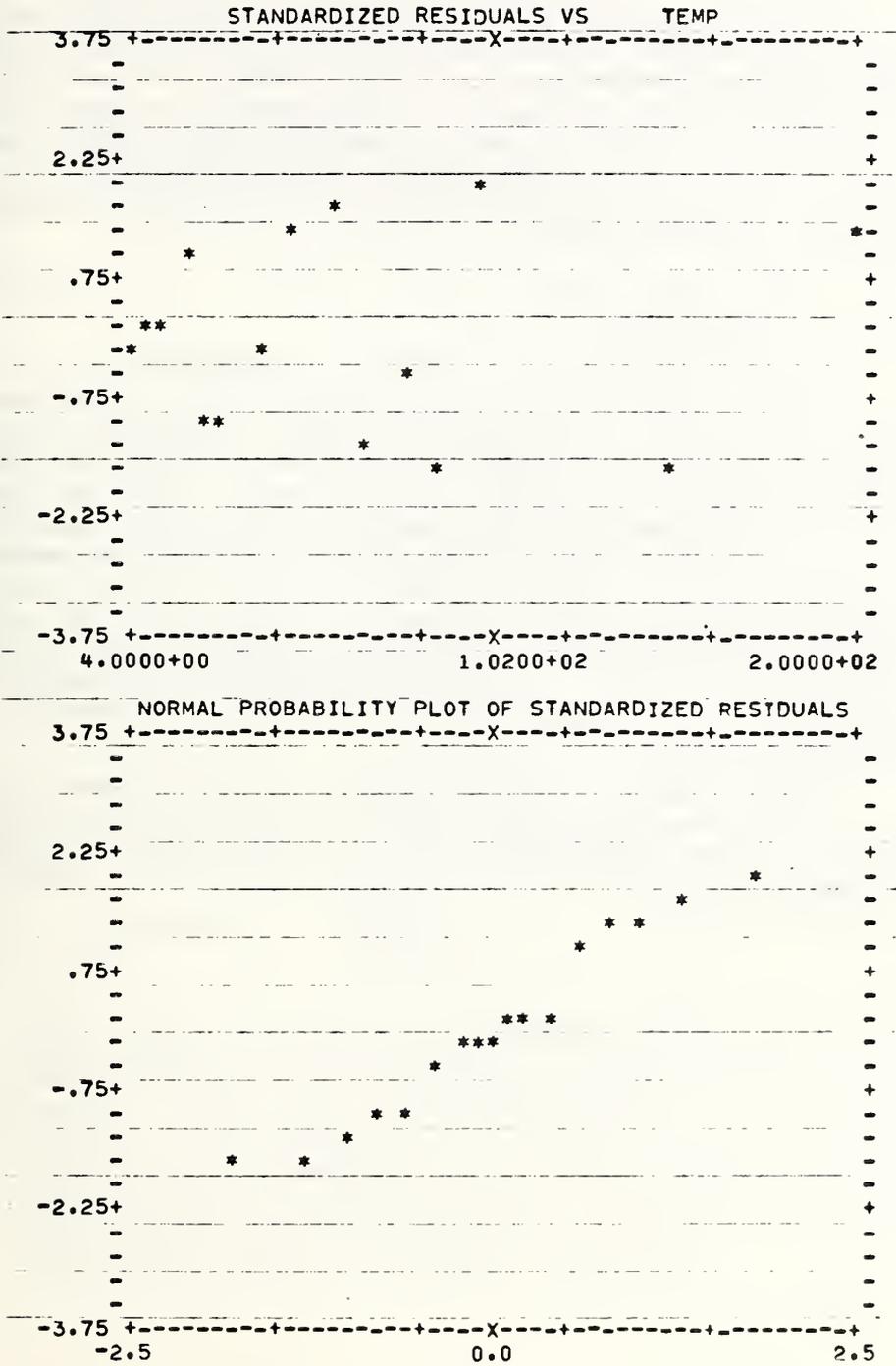


Figure 5.15g. Here we see the format of the graphical output when it is directed to a narrow terminal.

## 5.16 GRAPH

This module generates bar graphs (histograms) corresponding to the numeric values of each record of the designated vectors. For each graph desired, the user must specify: the LABEL of the vector to be graphed; the symbols to be used in the graph; and which vector is to supply the information for the stub of the graph. As many as 10 different graphs can be generated in one pass through the file and each graph can consist of different symbols. When no stub is necessary or desired, the third item in response to any of the requests:

```
*TYPE LABEL, SYMBOL(S), STUB ---> ?
```

must be the word NONE.

Figure 5.16a shows the instructions for producing three graphs of data stored in a small test file and a portion of the first graph. Subsequent figures show the appearance of the other two graphs. Note that when stubs are specified, the user can specify how many characters of the stub field should be printed. If this number exceeds the defined width of the data vector used for the stub, the values found are padded out with blanks. If no instruction is given regarding the length of the stub field, the module assigns 12 characters to it. When more than one graph is requested, it may be useful to have them scaled uniformly. This module allows for such uniform scaling and formats the histograms to facilitate comparisons by starting them all in the same position on the print line. This feature is especially useful when the graphs depict the same items of information for different fiscal years.

Additional features of this module are the options it provides for combining the graphs into a composite picture in which the adjacent bars may have different symbols. Figures 5.16c and 5.16d show the utility of this feature of the GRAPH module. Other graphical facilities are provided in the PLOT module.

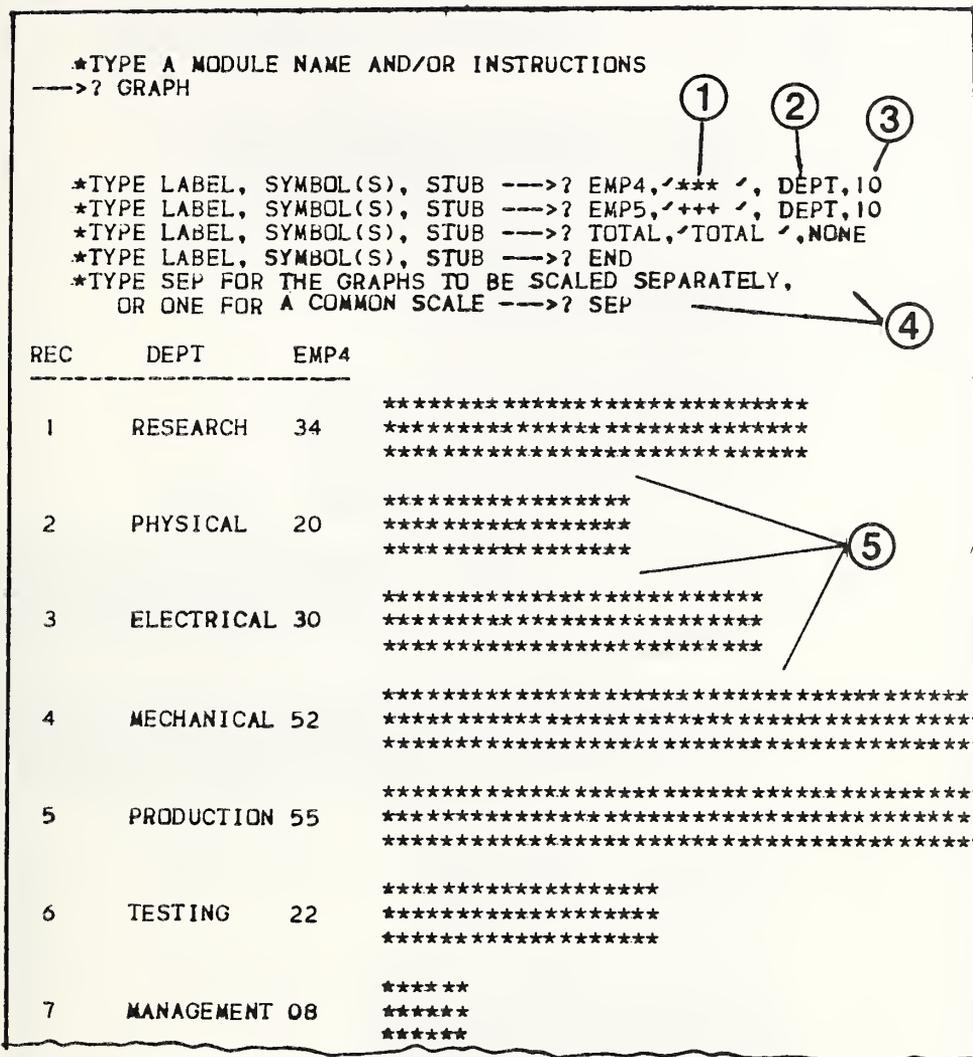


Figure 5.16a. Instructions for the generation of three graphs of data vectors EMP4 and EMP5 containing staffing levels for two fiscal years showing: 1) how the user specifies the symbols to be used in each histogram; 2) which data vector supplies the stub; 3) the number of characters to use for the stub; and finally 4) the instructions for the type of normalization. Note that the blank line between bars (5) comes from the space embedded in the symbols supplied in instruction (1) above. The second plot differs from this one only in the symbols used and that the values are taken from the vector EMP5 instead of EMP4.

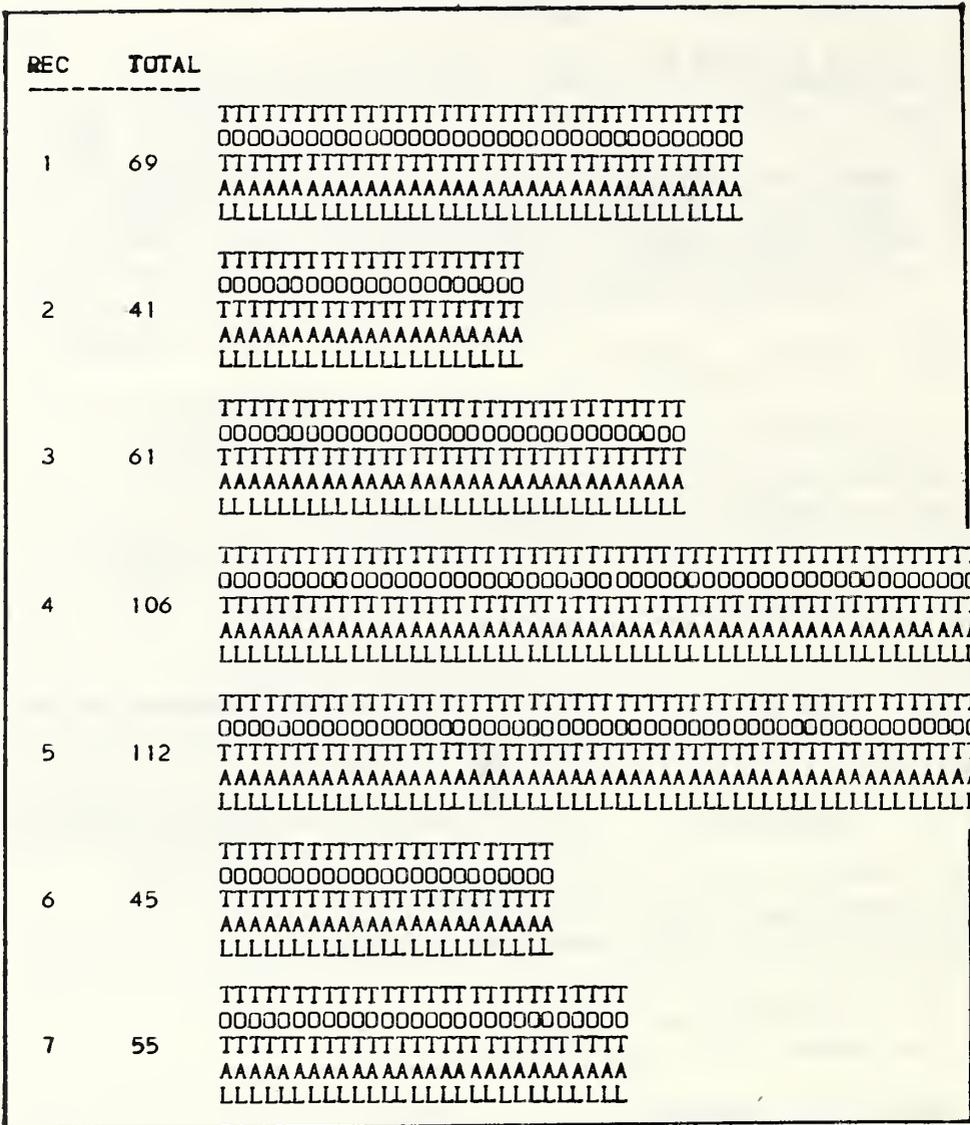


Figure 5.16b. In this second graph we see how the symbols used to produce the bars can also convey information. Here we have graphed the total of vectors EMP4 and EMP5. The number sequence on the left serves to identify the record for which the data are shown and can be used to correlate graphs of other data items in the same record. Had we asked for the graphs to be printed on a common scale, the bars would be shifted to the right to agree with the other graphs. Since we did not ask for a STUB, the graph is shifted to the left.

\*SHALL WE COMBINE - TYPE YES OR NO --->? YES  
 \*TYPE GRAPH NUMBERS TO BE COMBINED (IN DESIRED ORDER)

--->? 1,2,3 **①**

1	RESEARCH	34	***** ***** *****
1	RESEARCH	35	+++++ +++++ +++++
1		69	TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
2	PHYSICAL	20	.***** ***** *****
2	PHYSICAL	21	+++++ +++++ +++++
2		41	TTTTT TTTTTTTTTTTT OOOOOOOOOOOOOOOOOO TTTTTT TTTTTTTTTT AAAAAA AAAAAA AAAAAA LLLLLLLLLLLLLLLLLL
3	ELECTRICAL	30	***** ***** *****
3	ELECTRICAL	31	+++++ +++++ +++++
3		61	TTTTTT TTTTTT TTTTTT TTTTTT OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA LLLLLLLLLLLLLLLLLL LLLLLLLLLL

Figure 5.16c. Here we see how easy it is to prepare a composite graph of the individual histograms shown earlier. The order in which the bars are presented is determined by the answer supplied at (1) above. The next figure shows still another arrangement produced in the same run. It should be remembered that inclusion of totals in a composite plot compresses the scale and thereby deemphasizes the differences between the individual bars.

```

*SHALL WE COMBINE - TYPE YES OR NO -->? YES
*TYPE GRAPH NUMBERS TO BE COMBINED (IN DESIRED ORDER) -->? 3,1,2,3

1          69          TTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
                   OOOOOOOOOOOOOOOOOOOOOOOOOOO
                   TTTTTTTTTTTT TTTTTT TTTTTT TTTTTT
                   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                   LLLLLLLLLLLL LLL LLLLLLLLLLLLLLLLLL

1  RESEARCH C 34     *****
                   *****
                   *****

1  RESEARCH C 35     ++++++
                   ++++++
                   ++++++

1          69          TTTT TTTTTT TTTTTT TTTTTT TTTTTT
                   OOOOOOOOOOOOOOOOOOOOOOOOOOO
                   TTTTTT TTTTTT TTTTTT TTTTTT TTTT
                   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                   LLLLLLLLL LLLLLL LLLLLL LLLLLL LLLL

2          41          TTTTTTTTTTTT TTTTT
                   OOOOOOOOOOOOOOOOOOO
                   TTTTTT TTTTTT TTTTTT
                   AAAAAA AAAAAA AAAAAA
                   LLLLLLLLL LLLLL LLLL

2  PHYSICAL S 20     *****
                   *****
                   *****

2  PHYSICAL S 21     ++++++
                   ++++++
                   ++++++

2          41          TTTTT TTTTTT TTTTTT
                   OOOOOOOOOOOOOOOOOOO
                   TTTTTT TTTTTT TTTTTT
                   AAAAAA AAAAAA AAAAAA
                   LLLLLLLLL LLLLL LLLL

```

Figure 5.16d. Until the user answers NO to the first question, the module will continue to provide other variants. Here we chose to repeat the total column. Such repetition would be more useful if eight or more graphs were combined.

## 5.17 KWOC

One of the early (circa 1960) applications of computers to bibliographic work involved a so-called Key Word in Context (KWIC) index suggested and developed by the late H. P. Luhn. In this form of the permuted title index, each important word in all of the titles in a list is displayed in alphabetic order and arranged as shown in figure 5.17a, so that the word in the center column appears *in context* in its proper position in the title. In this format, the titles span the indexed word and space considerations often dictate the truncation of the titles either at the beginning or the end. That information is, however, not lost since the missing segment can be picked up again on the basis of another word which is closer to the beginning or end of the title.

An alternate approach to such a permuted title index is to place each important (indexable) word in front of the title and follow it after a suitable number of spaces or on the next line with the entire title in its original order. In this form the indexed word is alphabetized out of context, hence *Key Word Out of Context* (KWOC).

If, however, the indexed word in each line is accented within the title either by being printed in italics, in bold face or set off by asterisks or plus signs, one can have the advantages of both the KWIC and the KWOC without the drawbacks of the former. Figure 5.17b shows the appearance of the on-line output of an index produced by this module. The dialogue between the module and the user is shown in figure 5.17e, et seq.

It will help the user to understand the reasons for the questions that are asked by the module if we digress to explain that many of the anticipated applications of this module have as their final objective the production of a phototypeset printed volume. For that purpose, the system must be able to insert into the text stream arbitrary character strings to identify which portions of the text are to be set in bold face or italics, etc. Since Omnidata handles only upper case characters at present, this module will recognize case shift symbols and transmit them along with the text, but ignore them when required as in figure 5.17b. Other features of this module will emerge as we go through the dialogue step by step for applications of the KWOC module to generate indexes for various publication series produced by the Office of Standard Reference Data (OSRD).

Figure 5.17c shows how the bibliographic information was keypunched for articles in the *Journal of Physical and Chemical Reference Data (JPCRD)*. A short X BASIC program was written to convert this card-image file to fixed-field format required by the DEFINE module. The equal signs were inserted in the text so that the file could be used later to produce indexes in upper and lower case. Figure 5.17d shows such an index produced by a text-formatting program which read the file produced by this module.

REV - ROT	TITLE WORD INDEX	AMS 64	232
PROBLEMS	A REVIEW OF THE LITERATURE ON A CLASS DF COVERAGE	AMS 64	232
	BOOK REVIEWS. 10 YEAR INDEX (1959-1968)	TECH 69	223
	SPECTRAL EVALUATION OF BLS AND CENSUS REVISED SEASONAL ADJUSTMENT PROCEDURES	JASA 68	472
	ON THE STATISTICAL DISCREPANCY IN THE REVISED UNITED STATES NATIONAL ACCOUNTS	JRSSB54	292
	TIZED DEVIATE FROM THE SAMPLE MEAN	JASA 66	1219
	DATA REVISIONS AND ECONOMIC FORECASTING	BIOKA56	449
	TRANSFORMATIONS, SOME EXAMPLES REVISITED	JASA 67	470
	QUERY PREFERENCE SCORES (REVISED)	TECH 69	23
	COMPUTERS. THE SECOND REVOLUTION IN STATISTICS (THE FIRST FISHER MEMORIAL	TECH 68	612
LECTURE)	EXISTENCE OF OPTIMAL STOPPING RULES FOR REWARDS RELATED TO S-SUB-N-OVER-N	BIOCS66	233
	FROM A NORMAL BIVARIATE POPULATION WITH CORRELATION RHO /TRIBUTION OF KENDALL'S TAU FOR SAMPLES OF FOUR	AMS 68	122B
	THE VARIANCE OF SPEARMAN'S RHO IN NORMAL SAMPLES	AMS 68	122B
	OF A MAXIMUM-LIKELIHOOD ESTIMATION PROPOSED BY RICHARDS	BIOKA61	19
	A COMPUTER PROGRAM FOR FITTING THE RICHARDS FUNCTION	SOME REMARKS ON A METHOD	JRSSB63 209
		BIOCS69	401
	ESTIMATES OF THE REGRESSION COEFFICIENT BASED ON KENDALL'S TAU	BIOKA55	417
	DISTRIBUTIONS OF KENDALL'S TAU BASED ON PARTIALLY ORDERED SYSTEMS	BIOKA63	538
LATE POPULATION WITH CORRELAT/	THE DISTRIBUTION OF KENDALL'S TAU FOR SAMPLES OF FOUR FROM A NORMAL BIVAR	JASA 57	33
RANKINGS	A MODIFICATION OF KENDALL'S TAU FOR THE CASE OF ARBITRARY TIES IN BOTH	JASA 57	33
	THE THIRD MOMENT OF KENDALL'S TAU IN NORMAL SAMPLES	BIOKA62	177
	A COMPUTER METHOD FOR CALCULATING KENDALL'S TAU WITH UNGROUPED DATA	JASA 66	436
A SIMILARITY BETWEEN GOODMAN AND KRUSKAL'S TAU AND KENDALL'S TAU. WITH A PARTIAL INTERPRETATION OF THE L	JASA 62	804	
FUNCTIONS	INTEGRAL KERNELS AND INVARIANT MEASURES FOR MARKOFF TRANSITION	AMS 65	517
	ON DISTRIBUTIONS FOR WHICH THE HARTLEY-KHAMIS SOLUTION OF THE MOMENT-PROBLEM IS EXACT	BIOKA51	74
	SOME REMARKS CONCERNING KHATRI'S RESULT ON QUADRATIC FORMS	BIOKA68	593
	ON CONVERGENCE OF THE KIEFER-WOLFOVITZ APPROXIMATION PROCEDURE	AMS 62	1227
CORR. 66 745	A CONTINUOUS KIEFER-WOLFOVITZ PROCEDURE FOR RANDOM PROCESSES.	AMS 67	1D31
	ERRORS OF THE THIRD KIND IN STATISTICAL CONSULTING	AMS 64	590
ACTORIAL EFFECTS AND THEIR CONNECTION WITH A SPECIAL KIND OF IRREGULAR FRACTIONAL PLANS OF FACTORIAL EXPR	JASA 63	497	
ENOMIN/	ROBUSTNESS OF THE F-TEST TO ERRORS OF BOTH KINDS AND THE CORRELATION BETWEEN THE NUMERATOR AND D	JASA 68	660
THE MATHEMATICAL ANALYSIS OF AN EPIDEMIC WITH TWO KINDS OF	SUSCEPTIBLES	BIOCS68	557
	NOTE ON A THEOREM OF KINGMAN AND A THEOREM DF CHUNG	AMS 66	1844

Figure 5.17a. Here we see the characteristics of a KWOC index showing the usual format. Note that the portion of the title that is missing at [A] can be picked up later in a number of places as at [B].



```

* * * * * =DIFFUSION =COEFFICIENTS
* 00101=GASEOUS =MARRERO, =T. =R.; =MASON, =E. =A.
* 00102=MARRERO, =T. =R.; =MASON, =E. =A.
* 00103=J=P=C=R=D1972 =VOL.1 =NO.1 =P.3
* 00104BINARY GAS MIXTURES; CRITICALLY EVALUATED DATA; DIFFUSION COEFFICIENTS;
* 00104GASES; TRANSPORT PROPERTIES
* 00105DIFFUSION COEFFICIENT; POTENTIAL ENERGY CURVES FOR ATOMS AND MOLECULES
* 00201=SELECTED =VALUES OF =CRITICAL =SUPERSATURATION FOR =NUCLEATION OF
* 00201=LIQUIDS FROM THE =VAPOR
* 00202=POUND, =G. =M.
* 00203=J=P=C=R=D1972 =VOL.1 =NO.1 =P.119
* 00204CRITICAL SUPERSATURATION; DATA EVALUATIONS; HOMOGENEOUS NUCLEATION; PHASE
* 00204CHANGE
* 00205CRITICAL SUPERSATURATION RATIO
* 00301=SELECTED =VALUES OF =EVAPORATION AND =CONDENSATION =COEFFICIENTS FOR
* 00301=SIMPLE =SUBSTANCES
* 00302=POUND, =G. =M.
* 00303=J=P=C=R=D1972 =VOL.1 =NO.1 =P.135
* 00304CONDENSATION COEFFICIENTS; DATA EVALUATION; EVAPORATION COEFFICIENTS; PHASE
* 00304CHANGE
* 00305EVAPORATION AND CONDENSATION COEFFICIENTS
* 00401=ATLAS OF THE =OBSERVED =ABSORPTION =SPECTRUM OF =CARBON =MONOXIDE =BETWEEN
* 004011060 AND 1900 ANGSTROM
* 00402=TILFORD, =S. =G.; =SIMMONS, =J. =D.
* 00403=J=P=C=R=D1972 =VOL.1 =NO.1 =P.147
* 00404ABSORPTION SPECTRA; CARBON MONOXIDE; ELECTRONIC TRANSITIONS; IDENTIFICATION*
* 00404ATLAS; POTENTIAL ENERGY CURVES; ROTATIONAL AND VIBRATIONAL CONSTANTS
* * * * *

```

Figure 5.17c. A listing of card images of the original free-field reference file from which the Omnidata file was built to be processed by the KWOC module. Note the use of equal signs to denote upper case characters and the trailing digits (1,2,3,4,5) in the sequence numbers which identify the cards as containing: titles, authors, citations, keyword phrases, and property index terms. Each of these fields is a candidate for indexing by this module. Figure 5.17g shows the first logical record from this file as it is defined in the Omnidata file.

**Absorption**

Atlas of the Observed Absorption Spectrum of Carbon Monoxide Between 1060 and 1900 @Vol.1 No.1 P.147  
Atlas of the Absorption Spectrum of Nitric Oxide (NO) between 1420 and 1250 @Vol.5 No.2 P.309

**Acetaldehyde**

Microwave Spectra of Molecules of Astrophysical Interest - IX. Acetaldehyde Vol.5 No.1 P.53

**Acid**

Microwave Spectra of Molecules of Astrophysical Interest - X. Isocyanic Acid Vol.5 No.1 P.79

**Actinide**

Ground Levels and Ionization Potentials for Lanthanide and Actinide Atoms and Ions Vol.3 No.3 P.771

**Activity**

Osmotic Coefficients and Mean Activity Coefficients of Uni-univalent - Electrolytes in Water at 25 deg C Vol.1 No.4 P.1047

**Adjustment**

The 1973 Least-Squares Adjustment of the Fundamental Constants Vol.2 No.4 P.663

**Alkali**

Refractive Index of Alkali Halides and Its Wavelength and Temperature Derivatives Vol.5 No.2 P.329

**Alkoxy**

A Critical Review of H-Atom Transfer in the Liquid Phase: Chlorine Atom, Alkyl, Trichloromethyl, Alkoxy, and Alkylperoxy Radicals Vol.3 No.4 P.937

**Alkyl**

A Critical Review of H-Atom Transfer in the Liquid Phase: Chlorine Atom, Alkyl, Trichloromethyl, Alkoxy, and Alkylperoxy Radicals Vol.3 No.4 P.937

**Alkylperoxy**

A Critical Review of H-Atom Transfer in the Liquid Phase: Chlorine Atom, Alkyl, Trichloromethyl, Alkoxy, and Alkylperoxy Radicals Vol.3 No.4 P.937

**Allowed**

Atomic Transition Probabilities for Scandium and Titanium - (A Critical Data Compilation of Allowed Lines) Vol.4 No.2 P.263

**Alloys**

Elastic Properties of Metals and Alloys, I. Iron, Nickel, and Iron-Nickel Alloys Vol.2 No.3 P.531  
Diffusion in Copper and Copper Alloys - Part I. Volume and Surface Self-Diffusion in Copper Vol.2 No.3 P.643  
Diffusion in Copper and Copper Alloys - Part II. Copper-Silver and Copper-Gold Systems Vol.3 No.2 P.527

Figure 5.17d. A portion of an index produced by a text formatting program from a file generated by the KWOC module. When this file is processed further for phototypesetting, the word out-of-context on the left will appear in bold face and the word in-context will appear in italics.

An examination of figure 5.17c shows that the file contains a list of keywords as well as a list of properties under which the paper can be indexed. These data fields, as well as the title and author fields, are candidates for permutation. The module allows for mixing the result of permutations on words (or phrases) on two data fields at a time. The consequences and options pertaining to such mixed applications are illustrated later in this section.

The dialogue which produced the KWOC index shown in figure 5.17b is illustrated in figure 5.17e, where we call the KWOC module at [A] while setting the line width to 80 characters via the global WIDTH command. At [B] and [C] we specify that it is the words in the TITLE field on which this module should operate (permute). If we had chosen to prepare a permuted keyword (actually key phrase) index, we would have supplied a *semicolon* as a word separator instead of a *space* in response to the question at [C].

Since we wish to permute only the words in the TITLE field, we respond at [D] with NONE. Had we specified one or two secondary fields as we do later (see fig. 5.17h), the module would have requested additional "word" separators.

The CITATION field is specified at [E] as the field in the file labeled ID. As this index is to papers in a single journal, only volume, number, and page are given in the citation. The journal designation (JPCRD) is contained in the file as can be seen from figure 5.17d. If this file were combined with references in other journals, the journal designations would be included in the ID field when the file was defined originally. Alternatively, the journal field can be concatenated with the rest of the citation via the CONCAT module, prior to entering the KWOC module.

Next we must supply the module with information to insert certain characters or character strings to highlight the word out of context, as at [F], and the word in context, as at [G]. If we refer back to figure 5.17b we see that the word in context has indeed been bracketted by asterisks, while the word out of context has not been embedded between the designated plus signs. This is as it should be since the plus signs or other symbols indicated at [F] are needed only by the typesetting program (when the word is to appear in bold face type). Thus the file which this module creates does indeed contain the designated plus signs, but the program suppresses them in the printout.

Two other pieces of information are required before the module addresses itself to the question of which trivial words are to be excluded from the index. These inputs are requested at [H] for the symbol (or string) to separate the citation field from the title field, and at [I], for the shift symbol. The former is *inserted* into the output stream, while the latter is ignored in the output but retained in the file to be used in subsequent processing for computerized typesetting.

```

* * * * *
* FILE FTJPCRD CONTAINS 11 DATA ITEMS FOR 84 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->?
* KWOC
(A) *INPUT LABEL FOR WHICH KWOC INDEX IS DESIRED --->?
* TITLE
* *TYPE WORD SEPARATOR --->? (C) (B)
* , ,
* *TYPE ANY SECONDARY FIELD --->?
* NONE (E) (D) (F)
* *TYPE CITATION FIELD --->?
* ID
* *TYPE SYMBOLS TO HIGHLIGHT KEYWORD OUT OF CONTEXT --->?
* +,+
* *TYPE SYMBOLS TO HIGHLIGHT KEYWORD IN CONTEXT --->?
* * ,
* *TYPE TITLE TERMINATOR --->? (H)
* , -
* *TYPE SHIFT SYMBOL --->? (I) (G)
* =
* PRESENTLY, THE STOP LIST INCLUDES: (J)
* A AN AND AS AT
* BUT BY FOR FROM IN
* INTO OF ON OR THE
* TO WITH - BETWEEN ITS
* PART SELECTED THEIR THROUGH ONE
* TWO THREE FOUR FIVE SIX
* SEVEN EIGHT NINE TEN I
* II III IV V VI
* VII VIII IX X TWENTY-EIGHT
* *TYPE ANY WORDS YOU WISH TO DELETE FROM THIS LIST --->?
* NONE
* *TYPE ANY WORDS YOU WISH TO ADD TO THIS LIST --->
* NONE
* *TYPE NAME FOR KWOC INDEX ---> (L) (K)
* YES
* *HOW WIDE --->?
* 72
* * * * *

```

Figure 5.17e. Here we see the dialogue with the KWOC module that produced the index shown in figure 5.17b. See the text for comments on the marked places.

```

* * * * *
* AB INITIO CALCULATIONS      BINDING ENERGIES IN ATOMIC NEGATIVE IONS
*
* ABSORPTION SPECTRA        - VOL.4 NO.3 P.539
*
* ATLAS OF THE OBSERVED ABSORPTION SPECTRUM
* OF CARBON MONOXIDE BETWEEN 1060 AND
* 1900 A ANGSTROM - VOL.1 NO.1 P.147
*
* ABSORPTION SPECTRUM      ATLAS OF THE ABSORPTION SPECTRUM OF
*
* NITRIC OXIDE (NO) BETWEEN 1420 AND
* 1250 A ANGSTROM - VOL.5 NO.2 P.309
*
* ACTINIDE ELEMENTS        GROUND LEVELS AND IONIZATION POTENTIALS
*
* FOR LANTHANIDE AND ACTINIDE ATOMS AND
* IONS - VOL.3 NO.3 P.771
*
* ACTIVATION ENERGIES      EVALUATED CHEMICAL KINETIC RATE CONSTANTS
*
* FOR VARIOUS GAS-PHASE REACTIONS -
* VOL.2 NO.1 P.25
*
* ACTIVATION ENERGY       A CRITICAL REVIEW OF THE GAS-PHASE
*
* REACTION KINETICS OF THE HYDROXYL
* RADICAL - VOL.1 NO.2 P.535
*
* ACTIVITY COEFFICIENTS    OSMOTIC COEFFICIENTS AND MEAN ACTIVITY
*
* COEFFICIENTS OF UNI-UNIVALENT -
* ELECTROLYTES IN WATER AT 25 DEG C -
* VOL.1 NO.4 P.1047
*
* AIR CONSTITUENTS         SCALED EQUATION OF STATE PARAMETERS FOR
*
* GASES IN THE CRITICAL REGION - VOL.5
* NO.1 P.1
*
* ALKALI HALIDES           REFRACTIVE INDEX OF ALKALI HALIDES AND
*
* ITS WAVELENGTH AND TEMPERATURE
* DERIVATIVES - VOL.5 NO.2 P.329
*
* * * * *

```

Figure 5.17f. A portion of an index of keyword phrases associated with the papers listed to the right. See figure 5.17h for the instructions to the KWOC module to produce this index.

An effective KWOC indexing system must have built into it instructions for ignoring trivial words (articles, prepositions, connectives, conjunctions, etc., and possibly even the cardinal and ordinal numbers). As it is important, for reasons of economy in processing, to keep this list to a minimum, we have programmed this module to print out the built-in "stop list" at [J] and allow the user to delete words at [K] or to add words at [L] as required in the application at hand.

In normal uses of KWIC and KWOC indexes the technique is applied to the words in a single data field (such as the title). This module has been designed to handle more complex indexing. It can index authors, keyword phrases, subject index terms, etc. It even has provision to mix in the alphabetized list to the left, words from the title, keywords (or phrases), or index terms. In such a mixed index a decision must be made on what to list to the right of a keyword. One could, for example, list all of the keywords associated with the document as is done in a permuted keyword index. With this module, the user has complete freedom to format the index. Thus in figure 5.17f, we see an index of keyword phrases in which the title and the ID field are listed.

Before we take up the KWOC instructions required to prepare this index, it will be useful to examine figure 5.17g. There we see how the file is defined and the labels assigned to the information in the record. It should be noted that the authors, keywords, and property index terms are separated by semicolons. Note also that the ID field at [B] is defined to include the three pieces of information at [A]. In other applications it would be logical to include the journal and the year in the ID field as well. In any case, such a concatenation can be achieved later, if needed, by using the CONCAT module.

In figure 5.17h we see a variation from the dialogue shown in figure 5.17e. First we indicate at [A] that the title field should only be listed, hence we are not asked to supply a word separator. At [B] we nominate the keywords as the secondary field to be indexed and supply at [C] the semicolon as the "word" separator. The rest of the dialogue continues in the usual fashion.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >ftjpcrd
* FILE FTJPCRD CONTAINS 11 DATA ITEMS FOR 84 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >labels
* THE FILE FTJPCRD CONTAINS DATA LABELED AS FOLLOWS:
* 8  AUTHORS          7  ID          2  JOURNAL          10  KEYWORDS
* 5  NO              1  NUM         6  PG              11  PROPERTIES
* 9  TITLE          4  VOL         3  YR
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >display
*
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >all
*
* RECORD NUMBER 1
* NUM - 001
* JOURNAL - =J=P=C=R=D
* YR - 1972
* VOL - =VOL.1
* NO - NO.1
* PG - =P.3
* ID - =VOL.1 =NO.1 =P.3
* AUTHORS - =MARRERO,=T.=T.;=MASON,=E.=A.
* TITLE - =GASEOUS =DIFFUSION =COEFFICIENTS
*
* KEYWORDS - BINARY GAS MIXTURES; CRITICALLY EVALUATED DATA;
* COEFFICIENTS; GASES; TRANSPOR
* T PROPERTIES
* PROPERTIES - DIFFUSION COEFFICIENT; POTENTIAL ENERGY CURVES
* ND MOLECULES
*   *MORE --->? >no
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >end
* * * * *

```

Figure 5.17g. A display of the first record of the Omnidata file FTJPCRD showing how the data items in the record were defined. In this file four fields are logical candidates for a permuted index: authors, title, keywords, and properties.

```

* * * * *
* *WHICH DATA BASE DO YOU WANT --->?
* FTJPCRD
* FILE FTJPCRD CONTAINS 11 DATA ITEMS FOR 84 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->?
* KWOC,WIDTH,132
* *INPUT LABEL, ETC. FOR PRIMARY FIELD --->?
* TITLE,LIST
* *TYPE ANY SECONDARY FIELD(S) TO BE INDEXED --->?
* KEYWORDS
* *TYPE SEPARATOR(S) --->?
* ',;'
* *TYPE CITATION FIELD --->?
* ID
* *TYPE SYMBOLS TO HIGHLIGHT KEYWORD OUT OF CONTEXT --->?
* +,+
* *TYPE SYMBOLS TO HIGHLIGHT KEYWORD IN CONTEXT --->?
* *
* *TYPE TITLE TERMINATOR --->?
* ',_'
* *TYPE SHIFT SYMBOL --->?
* =
* PRESENTLY, THE STOP LIST INCLUDES:
* A AN AND AS AT
* ~~~~~
* *TYPE ANY WORDS YOU WISH TO DELETE FROM THIS LIST --->?
* NONE
* *TYPE NAME FOR KWOC INDEX --->?
* TESTKWZ
* *DO YOU WISH TO PRINT KWOC INDEX --->?
* YES
* *HOW WIDE --->?
* 72
* * * * *

```

Figure 5.17h. Here we see the dialogue required to produce the keyword phrase index shown in figure 5.17f. See the text for comments on the marked portions. As this run was made in batch, the users responses are printed on a line below the question. As this run was made in batch, the users responses are printed on a line below the question.

\* \* \* \* \*

**Alberti, F.**  
Atlas of the Absorption Spectrum of Nitric Oxide (NO) between 1420 and 1250 Å—E. Miescher and F. Alberti. **5**, 309 (1976).

**Allen, C. B.**  
Molten Salts: Volume 4, Part 2, Chlorides and Mixtures. Electrical Conductance, Density, Viscosity, and Surface Tension Data—G. J. Janz, R. P. T. Tomkins, C. B. Allen, J. R. Downey, Jr., G. L. Gardner, U. Krebs, and S. K. Singer. **4**, 871 (1975).

**Allen, G. F.**  
An Analysis of Coexistence Curve Data for Several Binary Liquid Mixtures Near Their Critical Points—A. Stein and G. F. Allen. **2**, 443 (1973).

**Barnett, J. D.**  
High-Pressure Calibration. A Critical Review—D. L. Decker, W. A. Bassett, L. Merrill, H. T. Hall, and J. D. Barnett. **1**, 773 (1972).

**Bassett, W. A.**  
High-Pressure Calibration. A Critical Review—D. L. Decker, W. A. Bassett, L. Merrill, H. T. Hall, and J. D. Barnett. **1**, 773 (1972).

**Bauder, A.**  
Microwave Spectra of Molecules of Astrophysical Interest. IX. Acetaldehyde—A. Bauder, F. J. Lovas, and D. R. Johnson. **5**, 53 (1976).

\* \* \* \* \*

**Consolute point**  
An Analysis of Coexistence Curve Data for Several Binary Liquid Mixtures Near Their Critical Points—A. Stein and G. F. Allen. **2**, 443 (1973).

**Critical micelle concentration**  
(Abstract) Critical Micelle Concentrations of Aqueous Surfactant Systems (NSRDS-NBS-36)—Pasupati Mukerjee and Karol J. Mysels. **1**, 219 (1972).

**Critical supersaturation ratio**  
Selected Values of Critical Supersaturation for Nucleation of Liquids from the Vajor—G. M. Pound. **1**, 119 (1972).

**Critical temperature, pressure (see also Equation of state)**  
Analysis of Specific Heat Data in the Critical Region of Magnetic Solids—F. J. Cook. **2**, 11 (1973).  
An Analysis of Coexistence Curve Data for Several Binary Liquid Mixtures Near Their Critical Points—A. Stein and G. F. Allen. **2**, 443 (1973).

\* \* \* \* \*

Figure 5.17i. Examples of published indexes for which this module is able to prepare either a manuscript or a computer file for subsequent processing for computer-assisted typesetting.

## 5.18 PLAN

The Omnidata system was designed primarily to provide on-line interactive (conversational) data retrieval, data analysis and reporting facilities. As important as this mode of operation is, there are circumstances where the file operations or reports will be required routinely on a daily, weekly, or monthly basis. In such cases, it is advantageous to be able to run the search, the analysis, and the report generation from a previously formulated set of instructions. The PLAN module provides such a facility.

We will discuss the features of this module by illustrating first the building of a new plan. This is done by calling for the PLAN module which responds in the manner shown in figure 5.18a. The user's response to the first question contains the name of the plan, QUARTERLY in this instance, and the fact that it is a new one.

The writing of a proper plan depends upon the user being familiar with the exact order in which the modules ask for input. Thus, after giving the name of the file on line 1 and calling for the TALLY module on line 2, he must know that the TALLY module will require the names of the vectors to be tallied. These we supply on line 3. On line 4 we call the DISTRIBUTE module and give instructions to it on line 5 to distribute the file on the REGION vector. In line 7 we give the instructions for the ANALYSIS, TANDEM operation requested in the line above.

This example should serve to emphasize that use of the PLAN module requires a detailed knowledge of the operations of the modules it calls upon. Novices should defer the use of this module until they have had some considerable experience with Omnidata.

Line 8 in figure 5.18a contains the instructions to run the plan immediately. At this stage, the module has already stored the instructions in lines 1 through 7 in the file QUARTERLY. It now replaces the word RUN by the word FINI before proceeding with the outlined operations. Had we added the letter P (for permanent) as one of the inputs when we responded to the first question, the file QUARTERLY would be catalogued permanently.

Let us now see how an earlier plan can be modified by calling up PLAN and responding with QUARTERLY, MODIFY. The module responds by printing out the existing plan as shown in figure 5.18b. Note that line 8 now reads FINI instead of RUN or CURRENT.

In the MODIFY mode, existing lines of the plan can be deleted or replaced and new instructions can be inserted. How this is done is shown and explained in figure 5.18b which also shows the modified plan which is printed out at the end of the operation.

There are actually three responses that can end a plan. They are: RUN, FINI, or CURRENT. RUN is used only to get an immediate output from the plan just generated. When the plan is ended with FINI it is not executed. It is simply stored away and catalogued permanently if the P option is used. If the P option is not used, there is little point in writing a plan that is not followed by a RUN instruction. When the plan is ended with CURRENT, this module ends the plan with FINI as before on the permanent file and executes the plan except that it uses the current file rather than the file named in the plan. The purpose of this feature is to allow the testing of the plan on a short test file rather than the larger file upon which it will normally operate.

Thus far we have shown how to write plans or modify plans and run them in the process. When plans already exist in catalogued files they can be utilized without calling the PLAN module. The instruction to use a particular plan can be supplied with the SIGN ON to the Omnidata system by following the account with PLAN XYZ, where XYZ is the plan name.

\* \* \* N O T E S \* \* \*

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >fsdemo ①
* FILE FSDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >plan
*   *TYPE PLAN NAME AND EITHER M(ODIFY) OR N(EW) --->
* ? quarterly,new
*   *TYPE COMMANDS --->
* 1 (FILE NAME) ? fndemo
* 2             ? tally
* 3             ? profs,pp,title,1 word,end
* 4             ? distribute
* 5             ? region
* 6             ? analysis,tandem
* 7             ? yob,eody,level
* 8             ? current ②
* PLAN QUARTERLY IS ASSIGNED TEMPORARILY
* CPU SEC IN PLAN = 2.5074 COST =
* CPU SEC = 7.5484 COST = TIME = 15:37:33 ③
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->tally
*   *TYPE LABELS (UP TO 3) TO BE TALLIED; FOLLOWED BY 'end'*
* --->profs,pp,title,1 word,end
*           TALLY OF PROFS
*
* PROFS      CUM %      % CUM FREQ FREQ
* -----
* 1          44.6      44.6   223 223 XXXXXXXXXXXXXXXXXXXXX*
* 2          61.2      16.6   306 83  XXXXXXXX
* 3          75.4      14.2   377 71  XXXXXXXX
*
* * * * *

```

Figure 5.18a. Here we see how a sequence of Omnidata operations is carried out from instructions supplied to the PLAN module as indicated on the opposite page. Note that since we selected the file FSDEMO (1) and ended the plan at line 8 with the word CURRENT (2), the plan is tested immediately on the file FSDEMO. Had we typed the word RUN instead of CURRENT in line 8, the plan would operate immediately on the file FNDEMO specified in line 1 of the plan. Note that beyond the point where costs are printed (3), the plan supplies all the instructions to the subsequent operations.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? fndemo
* FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? plan
*   *TYPE PLAN NAME AND EITHER M(ODIFY) OR N(EW) --->
* ? quarterly,modify
* PRESENTLY, YOUR PLAN IS AS FOLLOWS
* 1      FNDEMO
* 2      TALLY
* 3      PROFS,PP,TITLE,1 WORD,END
* 4      DISTRIBUTE
* 5      REGION
* 6      ANALYSIS,TANDEM
* 7      YOB,EODY,LEVEL
* 8      FINI
*   *TYPE # AND NEW COMMAND --->
* ? 3,profs,pp,end
* ? 3.1,display
* ? 3.3,all
* ? 3.5,no
* ? 3.7,end
* YOUR MODIFIED PLAN IS AS FOLLOWS:
* 1      FNDEMO
* 2      TALLY
* 3      PROFS,PP,END
* 4      DISPLAY
* 5      ALL
* 6      NO
* 7      END
* 8      DISTRIBUTE
* 9      REGION
* 10     ANALYSIS,TANDEM
* 11     YOB,EODY,LEVEL
* 12     FINI
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? plan
* * * * *

```

Figure 5.18b. Here we illustrate how an existing plan can be modified. Note: 1) the current version of the plan with the word FINI substituted (2) for CURRENT in line 8; 3) how we change a line in the plan; 4) how we interlineate instruction; and 5) how the modified plan looks.

## 5.19 PLOT

We think it not inappropriate to introduce this module by paraphrasing an old cliché—namely one picture is worth a thousand numbers. In figure 5.19a we see a table of numbers and in figure 5.19b we see the result of plotting that data in OMNITAB II. This is obviously an extreme example of the relative information content between the numerical and graphical displays. While the subsequent applications are less dramatic, they do illustrate the utility of a graphical analysis as an adjunct to the various numerical analysis facilities in the Omnidata system.

This module interfaces with the OMNITAB II system to provide a convenient and versatile facility for graphic displays of data. It puts all of the power of the OMNITAB II plotting facilities at the finger tips of the Omnidata users without burdening them to learn to write OMNITAB instructions. In this regard the Omnidata PLOT module represents a conversational front end to the OMNITAB II PLOT instructions, as it writes the appropriate OMNITAB instructions from information solicited from the Omnidata user.

The coupling between the two systems is made possible by the fact that the UNIVAC EXEC 8 system permits both Omnidata and OMNITAB II to operate on data files from instruction files on mass storage. Furthermore, the OMNITAB II instructions are systematic enough to permit Omnidata to generate them from information solicited from the user.

Plots produced by this module come in two sizes. One, produced by the OMNITAB II PAGE PLOT instruction, fits on an 8½x11 inch sheet. The other, produced by the OMNITAB II PLOT instruction, fits on an 11x12 inch sheet. Figure 5.19c is an example of a PAGE PLOT. On either plot it is possible to plot as many as five dependent variables (as ordinates) against one common independent variable (as abscissa). Figure 5.19d is a reduction of a full-size plot showing that it is possible to label both the horizontal and vertical axes. When two or more data points coincide, the number of such coincident points is printed instead of the symbols (., \*, +, comma, or -).

In order to achieve an interface with the OMNITAB II system, this module must do the following:

- a) transfer a number of designated numeric data vectors to either a permanent or temporary file;
- b) write a number of OMNITAB instructions to carry out the varied plotting operations;
- c) allow users to supply their own OMNITAB instruction both before and after the plot instructions;
- d) display the OMNITAB instructions it writes and give the user an opportunity to modify them or add to them; and,
- e) initiate an OMNITAB II run.



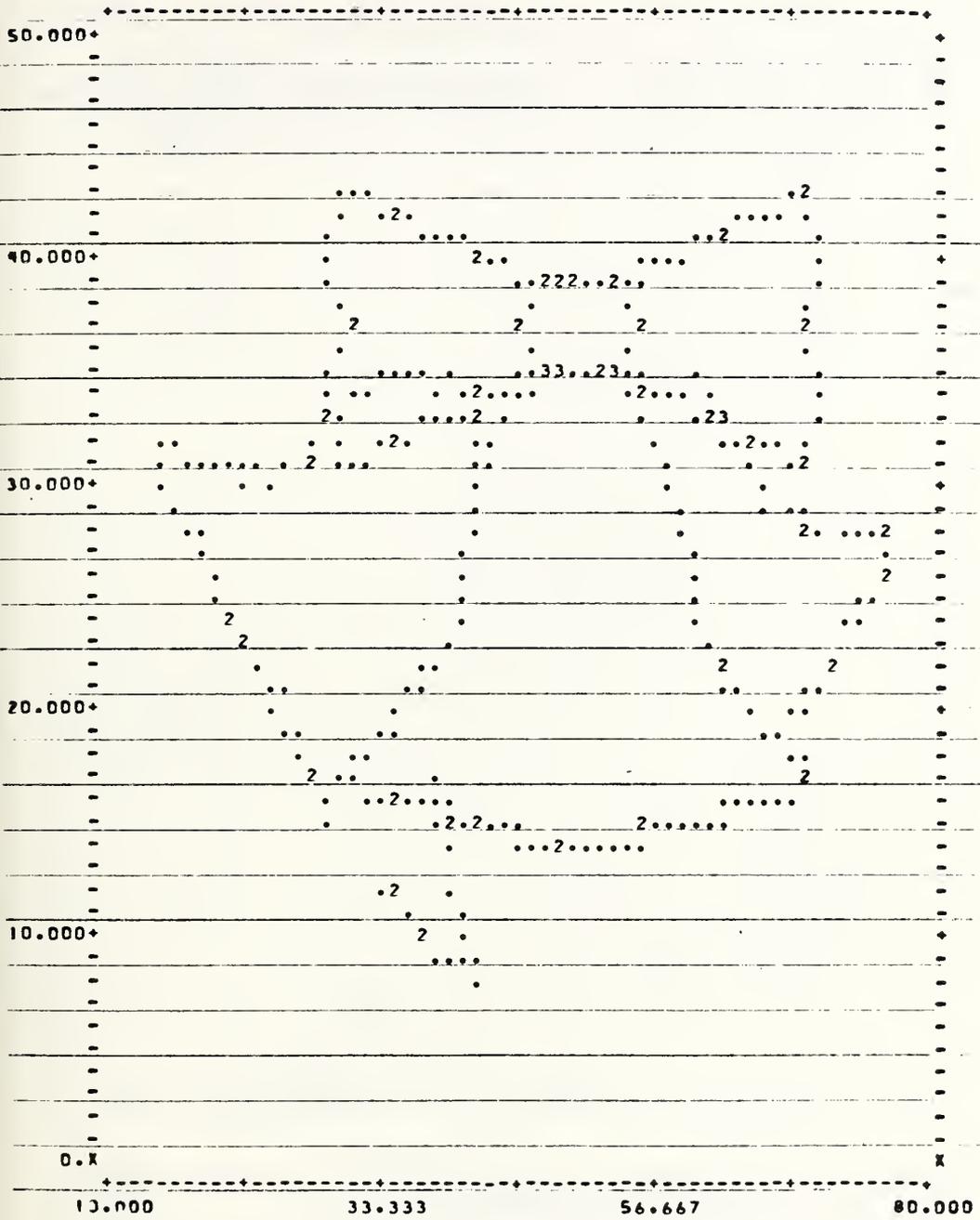


Figure 5.19b. A portion of a Christmas greeting produced by the OMNITAB plotting facilities from data points, some of which are shown in the previous figure.

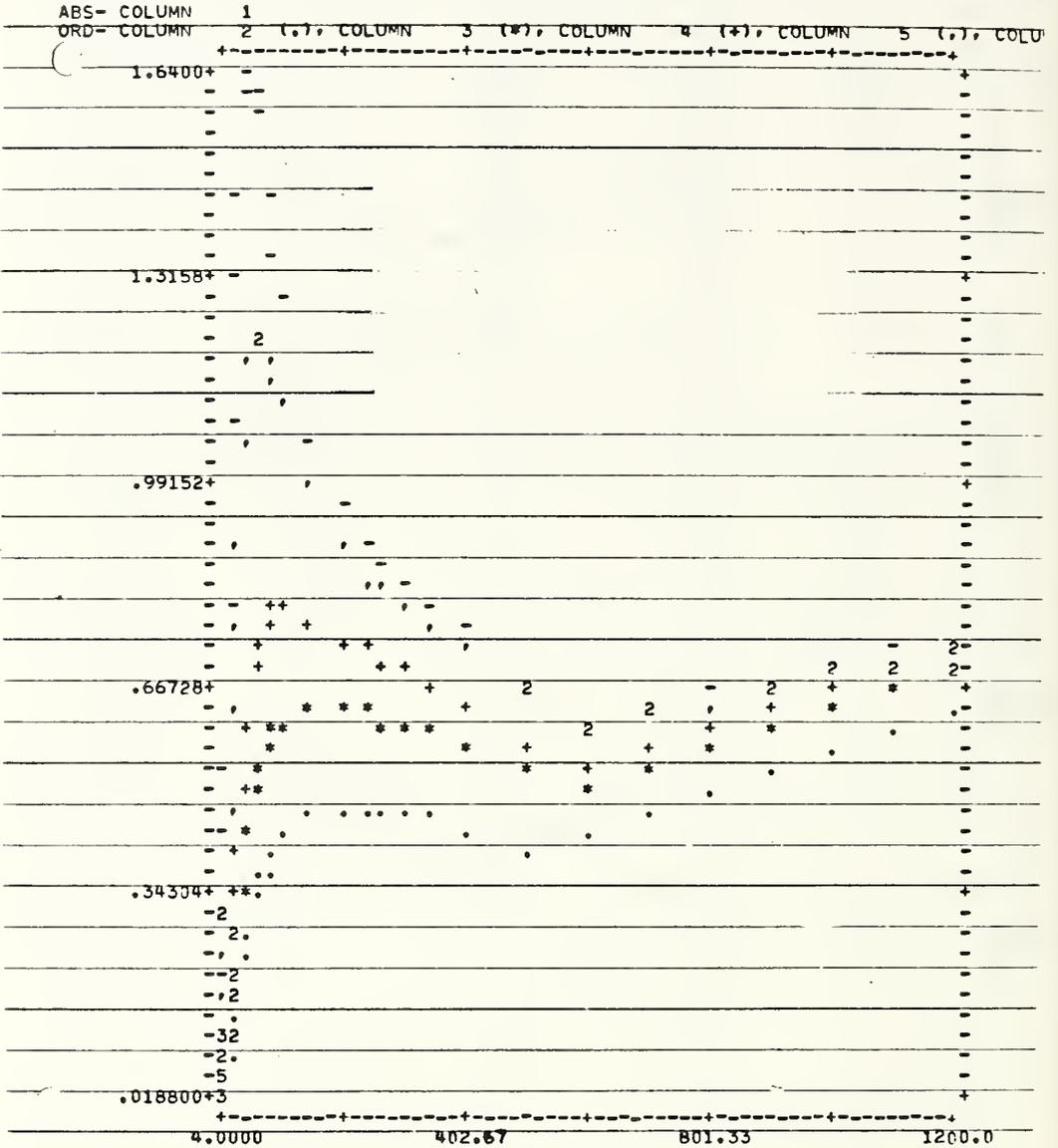


Figure 5.19c. An example of a typical OMNITAB II page plot showing how the five curves are presented and identified.

SALARY VERSUS AGE

ABS- CAGE  
ORD- PAY (.),

POINTS PLOTTED 414 POINTS NOT PLOTTED - OUT OF BOUND 15

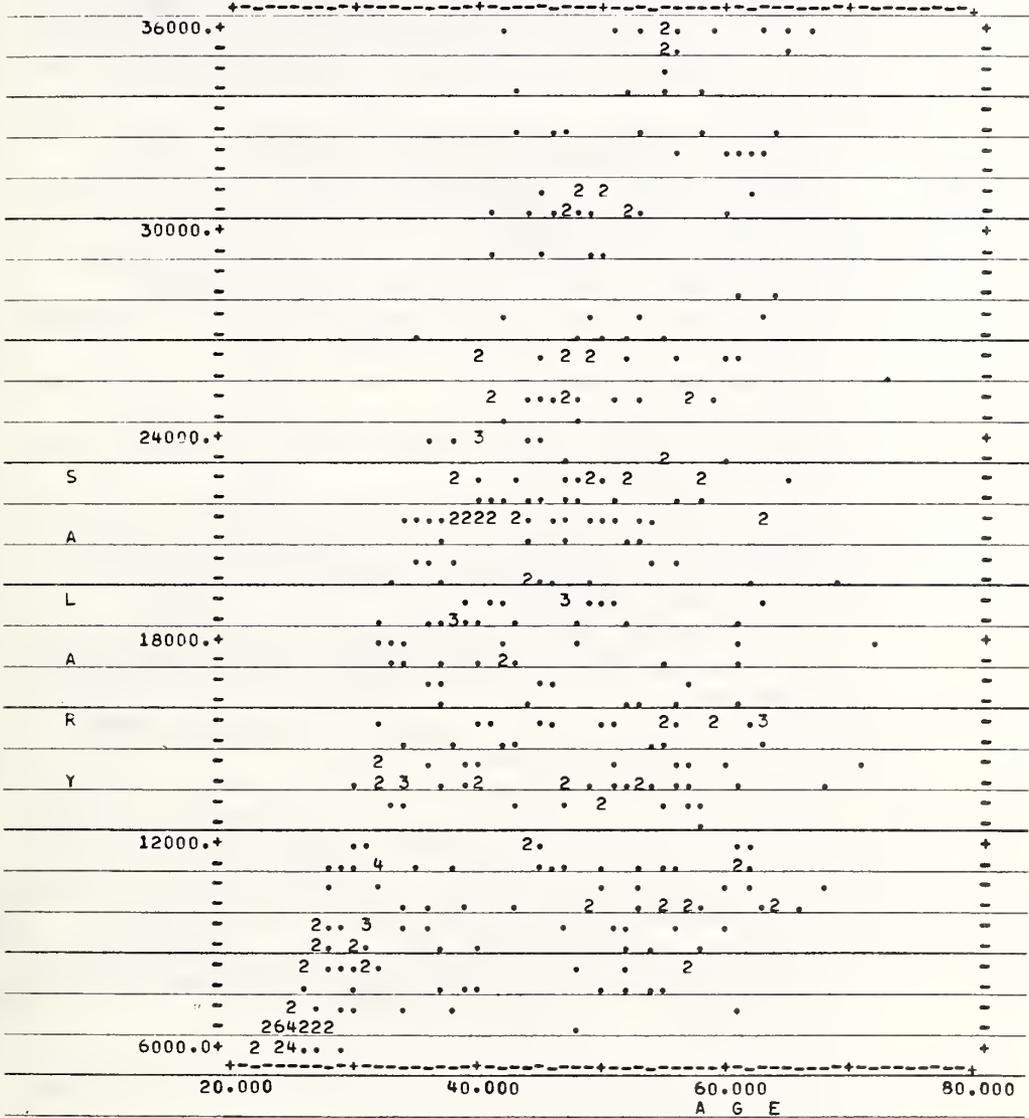


Figure 5.19d. A reduction of a plot produced by this module when the WIDTH is set to 80 characters. Note the numbers that replace the symbols when two or more points coincide and the facility for labeling the horizontal and vertical axes.

In figure 5.19e, we see how the user interacts with this module to plot certain data vectors from the file FPR75.

- \* The responses at [A], [B], and [C] transfer the 7 data vectors indicated at [C] to a permanently catalogued file PLOTTESTD. The name PLOTTESTD is derived by concatenating a 'D' (for data) to whatever the user elected to call the file—in this case PLOTTEST.
- \* At [D] the user is informed by the module as to how the OMNITAB worksheet has been dimensioned.
- \* At [E] the module reminds us where the data vectors are stored in the OMNITAB worksheet.
- \* The OMNITAB instructions at [F] are inserted here for illustrative purposes only. They do not affect the PLOT.
- \* At [G] we indicate the four data vectors which are to be plotted as ordinates on the scale specified at [H].
- \* At [I] we specify that the SEQ vector be used as the abscissa on the scale specified at [J].
- \* At [K] we supply a variety of legends for the title and for the horizontal and vertical axes.
- \* At [L] we have an opportunity to request more plots. Had we answered YES, the module would repeat the requests starting at position [F] to allow for further arithmetic manipulation before the next plotting operation.
- \* At [M] we are shown a finished list of OMNITAB II instructions produced by this module from the information supplied earlier.
- \* As we now see some errors, we respond affirmatively at [N].
- \* At [O] we replace instruction 12.
- \* At [P] we insert a line after line 18.
- \* At [Q] we retype line 10 to correct a misspelling of the word AND. This correction is really not necessary because OMNITAB ignores all but the first word in a command. After the first word, OMNITAB reads only the numbers in the instruction.
- \* The above comments apply also to the change made in instruction 11 at [R]. It, too, was really not necessary and was done to illustrate the correction procedure.
- \* At [S] we delete instruction 13.
- \* Finally, we request at [T] a new clean listing of the OMNITAB commands which are displayed at [U].
- \* At [V] we delete line 12 and do not ask for a new listing at [W].

At this point the Omnidata PLOT module initiates an OMNITAB II run. The plot in figure 5.19f, which is printed immediately on the terminal, is produced by the OMNITAB II system to which Omnidata has relinquished control.

```

* * * * *
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >plot
* *TYPE NAME FOR OMNITAB FILE --->? >plottest (A)
* *TYPE P(ERMANENT) OR T(EMPORARY) FOR FILE --->? >p
* *TYPE A TITLE FOR THIS RUN OR N(ONE)
* ? > none
* *TYPE LABELS TO BE TRANSFERRED TO OMNITAB
* (SEQ IS A LEGITIMATE ENTRY) --->
* ? > seq,age,pay,grade,div,deg,profs (B)
* *YOUR WORKSHEET IS DIMENSIONED FOR 201 ROWS BY 62 COLUMNS
* *THE DATA ARE STORED IN THE WORKSHEET AS FOLLOWS:
* COLUMN LABEL
* 1 SEQ
* 2 AGE
* 3 PAY
* 4 GRADE (E)
* 5 DIV
* 6 DEG
* 7 PROFS
* *INPUT ANY OMNITAB COMMANDS YOU WANT PERFORMED
* BEFORE PLOT—TYPE END TO END --->
* ? >add 1. to 1 and put results in column 8
* ? >multiply 2. by 8 and put results in column 9
* ? >add 8 to 9 and put results in column 10
* ? >print column 1
* ? >'print columns 1,2,3,4,8'
* ? >'print columns 1,2,3,4,8'
* ? >end
* *TYPE UP TO 5 COLUMNS TO BE PLOTTED ON ORDINATE --->
* ? >grade,div,deg,profs
* (G) *TYPE SCALE FOR VERTICAL AXIS (E.G. 0.0 to 60.0 OR D(DEFAULT) --->
* ? >0.0 to 18.0
* (H) *TYPE COLUMN TO BE PLOTTED ON ABSCISSA --->? >seq
* *TYPE SCALE FOR HORIZONTAL AXIS OR D(DEFAULT) ---> (I)
* ? >1.0 to 75.0
* *TYPE TITLE LINE ONE ---> (J)
* * * * *

```

Figure 5.19e. Here we see the user's dialogue with the PLOT module to transfer certain data vectors from an Omnidata file and to set up an OMNITAB II run to produce the plot shown in figure 5.19f. See the text for comments on the marked portions.

```

* * * * *
* *TYPE TITLE LINE ONE --->
* ? >' test of the plot module'
* *TYPE TITLE LINE TWO OR N(ONE) ---> (K)
* ? >none
* *TYPE NONE OR UP TO 60 CHARACTERS TO BE PRINTED HORIZONTALLY --->
* ? >sequence numbers
* *TYPE NONE OR UP TO 51 CHARACTERS TO BE PRINTED VERTICALLY --->
* ? >various vectors
* *DO YOU WANT ANY MORE PLOTS --->? >no (L)
* FOLLOWING IS A LIST OF YOUR OMNITAB PLOT INSTRUCTIONS:
* 1 @ASG,A PLOTTESTD
* 2 @USE 7.,PLOTTESTD
* 3 @NBS*OMNITAB.
* 4 OMNITAB
* 5 DIMENSION THE WORKSHEET TO HAVE 201 ROWS AND 62 COLUMNS
* 6 FORMAT A (6F12.0/1F12.0)
* 7 READ TAPE A A INTO COLUMNS 1***7
* 8 LABEL SEQ, AGE, PAY, GRADE, DIV, DEG, PROFS
* 9 ADD 1. TO 1 AND PUT RESULTS IN COLUMN 8
* 10 MULTIPLY 2. BY 8 ANC PUT RESULTS IN COLUMN 9
* 11 ADD 8 TO 9 AND RESULTS IN COLUMN 10
* 12 PRINT COLUMN 1
* 13 NPRINT COLUMNS 1
* 14 PRINT COLUMNS 1,2,3,4,8
* 15 TITLE1 TEST OF THE PLOT MODULE
* 16 TITLE3
* 17 TITL EX SEQUENCE NUMBERS
* 18 TITL EY VARIOUS VECTORS
* 19 PAGE PLOT 4 5 6 7 VERTICAL 0.0 TO 18.0,VS COL 1 HORIZONTAL
* 1.0 TO 75.0
* 20 STOP
* *DO YOU WISH TO CHANGE ANYTHING --->? >yes (N)
* *TYPE LINE NUMBER, NEW OR CORRECTED LINE --->
* ? >12,print columns 1 2 3 4 8 (O)
* ? >18.5,print columns 1 4 5 6 7 (P)
* ? >10,multiply 2. by 8 and put results in column 9 (R)
* ? >11,add 8 to 9 and put results in column 10 (Q)
* ? >delete 13 (S)
* ? >end (T)
* *DO YOU WANT A CLEAN LISTING OF YOUR OMNITAB COMMANDS --->? >yes
* FOLLOWING IS A LIST OF YOUR OMNITAB PLOT INSTRUCTIONS:
* 1 @ASG,A PLOTTESTD
* * * * *

```

Figure 5.19e (continued). The D at the end of the first two instructions is appended to the file name supplied by the user to indicate that the file contains the data. This module also writes a file of instructions which, in this case, is called PLOTTESTI. See the text for a discussion of the other items.

```

* * * * *
* FOLLOWING IS A LIST OF YOUR OMNITAB PLOT INSTRUCTIONS:
* 1 @ASG,A PLOTTESTD
* 2 @USE 7.,PLOTTESTD
* 3 @NBS*OMNITAB.
* 4 OMNITAB
* 5 DIMENSION THE WORKSHEET TO HAVE 201 ROWS BY 62 COLUMNS
* 6 FORMAT A (6F12.0/1F12.0)
* 7 READ TAPE A A INTO COLUMNS 1***7
* 8 LABEL SEQ, AGE, PAY, GRADE, DIV, DEG, PROFS
* 9 ADD 1. TO 1 AND PUT RESULTS IN COLUMN 8
* 10 MULTIPLY 2. BY 8 AND PUT RESULTS IN COLUMN 9
* 11 ADD 8 TO 9 AND PUT RESULTS IN COLUMN 10
* 12 PRINT COLUMNS 1 2 3 4 8
* 13 PRINT COLUMNS 1,2,3,4,8
* 14 TITLE 1 TEST OF THE PLOT MODULE
* 15 TITLE3
* 16 TITLX S E Q U E N C E N U M B E R S
* 17 TITLY V A R I O U S V E C T O R S
* 18 PRINT COLUMNS 1 4 5 6 7
* 19 PAGE PLOT 4 5 6 7 VERTICAL 0.0 TO 18.0,VS COL 1 HORIZONTAL
* 1.0 TO 75.0
* 20 STOP
* *DO YOU WISH TO CHANGE ANYTHING --->? >yes
* *TYPE LINE NUMBER, NEW OR CORRECTED LINE --->
* ? >delete 12
* ? >end
* *DO YOU WANT A CLEAN LISTING OF YOUR OMNITAB COMMANDS --->? >no
* *** @@PROCESSING COMPLETE.***
*
* TIME : 6.422
*
* LSD XBASIC 15:37:01 15 MAR 77
*
* OMNITAB
*
* SEQ AGE PAY GRADE
* 1.0000000 46.000000 30486.000 15.000000
* 2.0000000 41.000000 30486.000 15.000000
* 3.0000000 43.000000 25398.000 15.000000

```

U

V

W

X

Figure 5.19e (concluded). At point [X] the OMNITAB II system has taken over and provides the subsequent output.

A few words are in order concerning the final OMNITAB instructions. In the first two lines are EXEC 8 instructions which assign the file containing the data transferred from the Omnidata file. It is this file which is read by OMNITAB II when it executes the READ TAPE A command in line 7. The second A in this command specifies the format indicated in the format statement in line 6.

The third line calls the OMNITAB program which executes the instructions in lines 3-20. The LABEL instruction in line 8 tells OMNITAB II to assign these names to data stored in columns 1, 2...7, respectively. Thus, although the PRINT and the PAGE PLOT instructions in lines 18 and 19 use column numbers, the results in figures 5.19e and 5.19f refer to the data by name rather than numbers. The labeling feature is available only in version 5.13 of the OMNITAB II system. Because it is not available in earlier versions, this module writes the plot instruction (as in line 19) in terms of column numbers. Thus this module will work equally well in earlier versions of OMNITAB after the LABEL instruction has been deleted as was done with instruction 13 at position [S].

The above illustrated OMNITAB II instructions contain only a single page plot because we had neglected to set the WIDTH command to 120. Had we done so, this module would have written two plotting instructions—one for a full-size plot via the PLOT command and another via the PAGE PLOT command.

In figure 5.19g we have a plot of years of government service versus years at NBS for 200 professionals chosen at random from a larger file. If all of these persons started their government career at NBS, the points would all fall on a straight line. The name PLOTTESTD is derived by concatenating a ' Points falling above the line represent a person with 1 or more years of federal service prior to joining NBS.

Our experience in data analysis has shown that in most applications one would wish to produce a number of plots. In that case there is strong reason to do it in the batch mode as it is both faster and cheaper. In such a case it is often wise to include one or more extra plots in which the ranges of the horizontal and vertical axes are not specified. This has often provided good insurance against poor judgment in selecting axes under pressure of the on-line tempo. In addition, if we had not allowed the system to choose its own axes in figure 5.19g we would not be alerted to the three data points which are in error.

The plotting facilities in OMNITAB II are much more extensive than we have shown here. The later versions of OMNITAB II have provision for:

- a) user control of plotting symbols including letters and numbers via the CPLOT instruction;
- b) user control of the over-all dimensions of the plots by specification of LENGTH and WIDTH; and
- c) producing smaller plots, either two per page via the TWOPLOTS instruction or four per page via the FOURPLOTS

The characteristics of and means for obtaining these varied plot formats are given in Hogben and Peavy [12].

OMNITAB

TEST OF THE PLOT MODULE

ABS- \ SEQ  
 ORD- GRADE (.), DIV (\*), DEG (+), PROFS (,),  
 POINTS PLOTTED 300 POINTS NOT PLOTTED - OUT OF BOUND 0

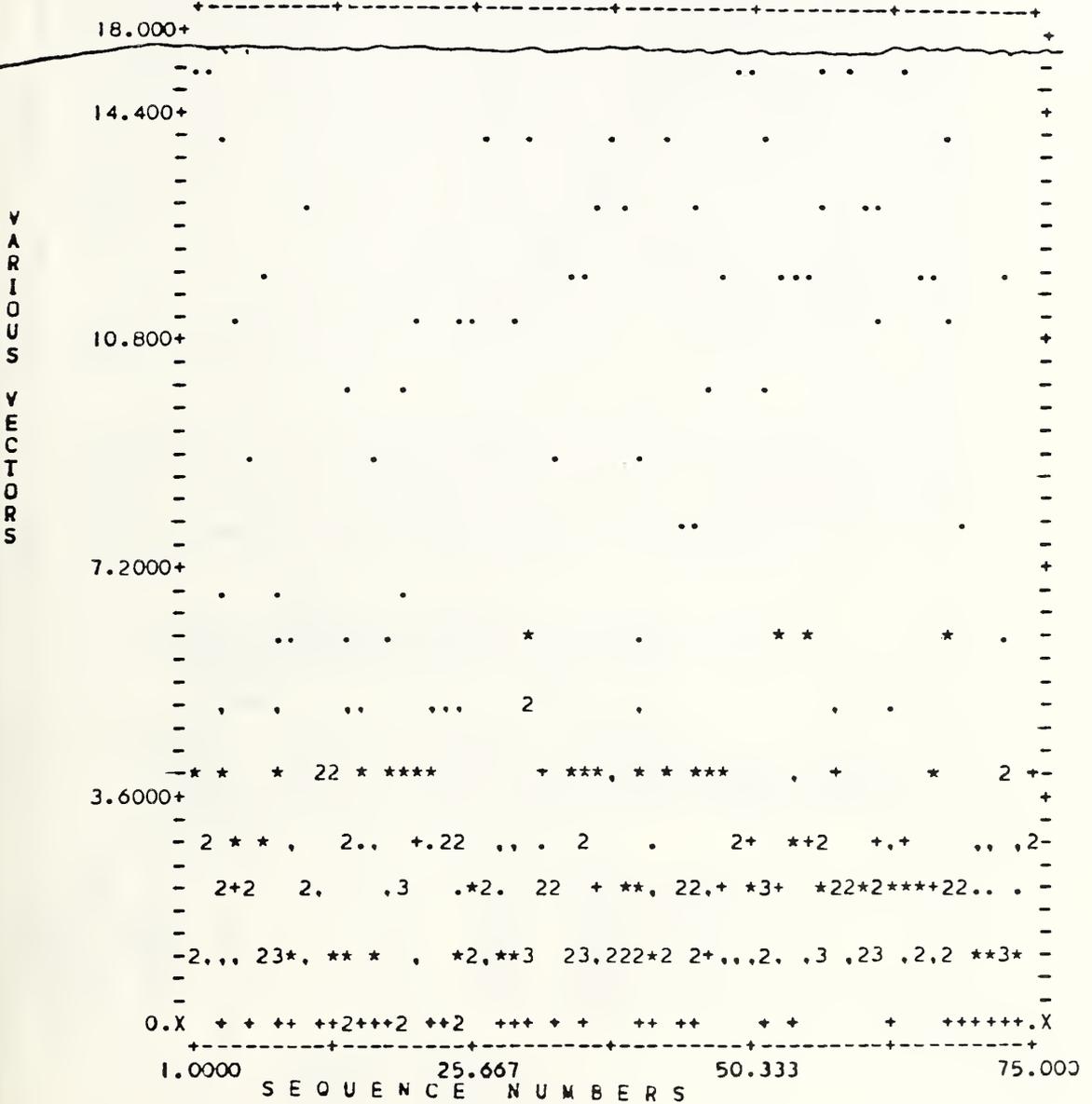


Figure 5.19f. Here we see a plot produced on-line by this module from the instructions shown in the previous figure. Note how the horizontal and vertical axes are labeled. This plot is more illustrative than informative.

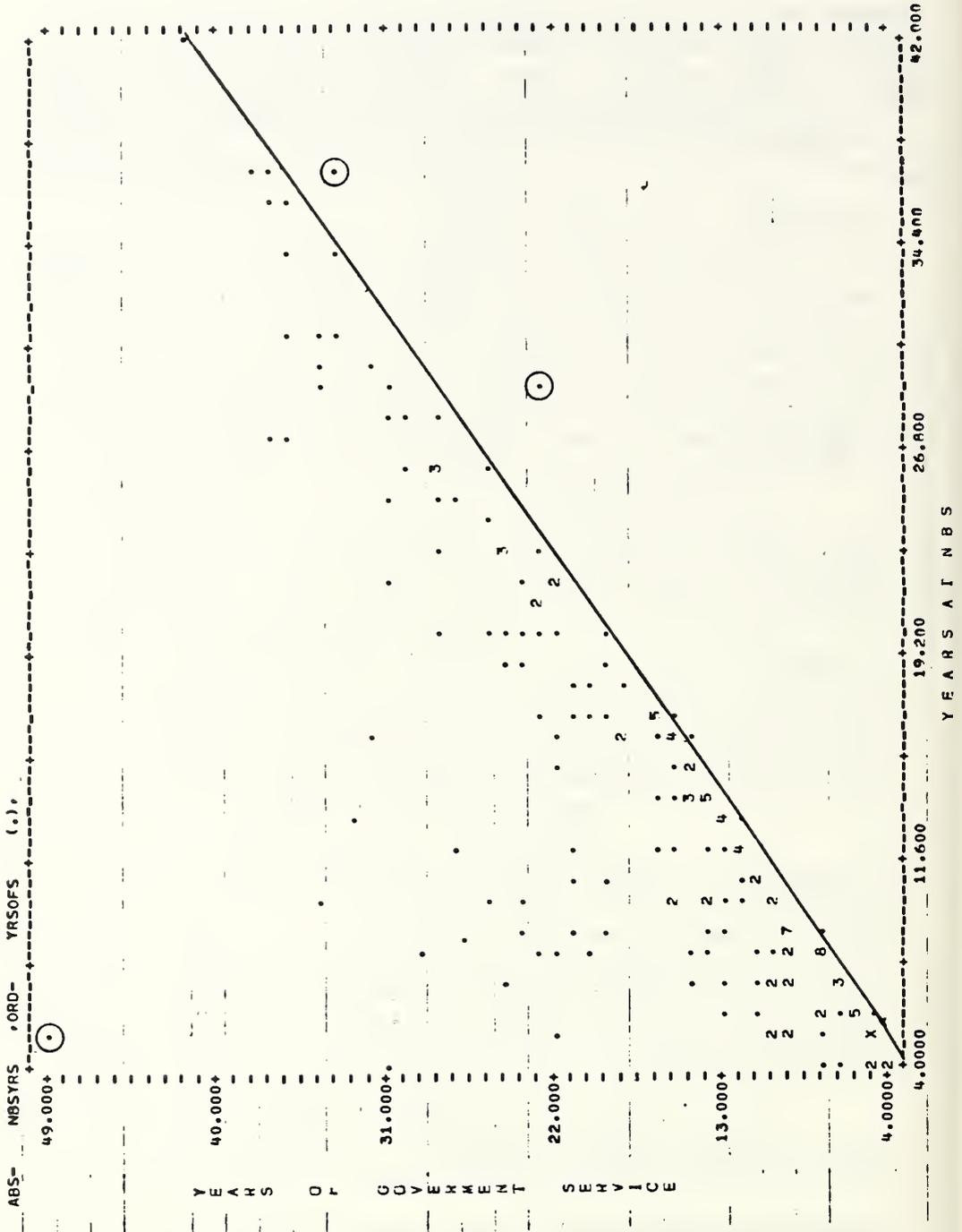


Figure 5.19g. This plot of years of government service versus years at NBS for 200 professionals chosen at random shows that recent hires had more prior government service than those in earlier years. The three circled points result from errors in the file.

## 5.20 RANDOM

In a variety of auditing operations it is important to be able to select, in a random manner, a sampling of records (logical or physical) from a larger data file. This module provides such a random sampling tool by writing a new file containing the desired number of randomly chosen records. The records are chosen via a list of random numbers generated and normalized to the number of entries in the file in question. Thus, if the file contained 1000 logical records and we wanted a sampling of 20%, the RANDOM module will be instructed to generate 200 unique random integers between 1 and 1000 and select from the large file logical records located in positions consonant with the generated list of integers. The module gets its information about the size of the sample from an input by the user. Information about the size (number of records) of the entire file is gotten automatically from the file itself. The records are written in the order in which they appeared on the original file. See figure 5.20a for a typical use of this module.

The further tasks in an auditing operation are also facilitated by OMNIDATA in that it has provisions for sorting the sample by organizational groups for possible interview or by building and room number for purposes of inventory. In the case of a physical inventory, the use of the features of the SEARCH module will allow 100% checking for high priced items; 50% checking for medium priced objects; etc.

```
* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? fndemo
*   FILE FNDemo CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   --->? random
*
* PRESENTLY, THERE ARE 500 RECORDS IN THE FILE.
*   *HOW MANY RECORDS DO YOU WANT SELECTED AT RANDOM --->? 50
*   *DO YOU WANT A LIST OF THE NUMBERS OF THE RECORDS SELECTED
*   --->? yes
*   THE 50 RECORDS SELECTED AT RANDOM ARE:
*   5 23 37 35 79 81 97 104 106
*   114 115 127 137 145 148 164 176
*   184 191 242 252 259 268 277 281
*   282 288 311 334 339 347 351 359
*   371 372 379 386 390 400 427 429
*   430 434 452 455 459 472 486 489
*   495
*
* CPU SEC IN RANDOM = 2.0806
* CPU SEC = 4.7286 TIM = 15:03:52
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   --->? stop
*   PROGRAM STOPPED.
*   TIME : 4.738
* * * * *
```

Figure 5.20a. A typical use of the RANDOM module to generate a 10% random sample of records from a larger file.

```

* * * * *
* PRESENTLY, THERE ARE 500 RECORDS IN THE FILE.
* *HOW MANY RECORDS DO YOU WANT SELECTED AT RANDOM --->? 50
* *DO YOU WANT A LIST OF THE NUMBERS OF THE RECORDS SELECTED
* --->? yes
*
* THE 50 RECORDS SELECTED AT RANDOM ARE:
* 17 19 31 47 51 75 79 100 112
* 124 142 151 162 187 213 219 223
* 224 238 247 274 278 286 287 312
* 319 326 338 354 364 366 383 384
* 385 392 398 400 405 414 416 425
* 440 443 457 466 468 477 490 495
* 496
*
* 10 ENTRIES COPIED 3.479
* 20 ENTRIES COPIED 3.4912
* 30 ENTRIES COPIED 3.506
* 40 ENTRIES COPIED 3.5186
* 50 ENTRIES COPIED 3.5324
*
* -----
* *WHICH DATA BASE DO YOU WANT --->? fndemo
* FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? random
*
* PRESENTLY, THERE ARE 500 RECORDS IN THE FILE.
* *HOW MANY RECORDS DO YOU WANT SELECTED AT RANDOM --->? 50
* *DO YOU WANT A LIST OF THE NUMBERS OF THE RECORDS SELECTED
* ② --->? no
*
* CPU SEC IN RANDOM = 2.0462
* CPU SEC = 4.9346 TIM= 15:06:58
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? stop
* PROGRAM STOPPED.
* * * * *

```

Figure 5.20b. A record of two more uses of RANDOM showing 1) the output with the MONITOR switch set at 10 records and 2) how this module finishes up when a printing of record numbers is not required.



The results of this operation are presented on four pages (see fig. 5.21b, et seq.) which contain the following.

On the first page we have a tabulation of:

- a) up to three of the independent variables (regressors)  $X_1$ ,  $X_2$ ,  $X_3$ , etc.;
- b) the dependent variable,  $Y$ , labeled as the response;
- c) the predicted response computed from this fitting operation;
- d) the standard deviation of each of the predicted responses;
- e) the residuals—the differences between the observed and the computed responses;
- f) the standardized residuals; and,
- g) the weights used in the fitting operation.

On the second page (fig. 5.21c) we have four plots showing:

- a) the standardized residuals versus row number;
- b) the standardized residuals versus predicted response;
- c) the standardized residuals versus the first column (the GNP Implicite Price Deflator); and,
- d) a normal probability plot of the standardized residuals.

On the third page (fig. 5.21d) we have two types of information of interest to the statistical analyst:

- a) the variance-covariance matrix, useful in assessing the collinearity of the data vectors fitted; and,
- b) the analysis of variance from which it is possible to assess the adequacy of the model.

On the fourth page (fig. 5.21e) we have the seven coefficients and their standard deviations, the ratio of each of the coefficients to its standard deviation, the number of digits in the coefficients that can be assumed to be correct, and finally, the standard deviation of the residuals. The above results are repeated for a second fit in which the last regressor variable has been eliminated.

---

\* \* \* N O T E S \* \* \*

LEAST SQUARES FIT OF RESPONSE, COLUMN 11,  
 AS A LINEAR FUNCTION OF 7 INDEPENDENT VARIABLES IN COLUMNS 1, 2, 3, 4, 5, 6, 7  
 USING 16 NON-ZERO WEIGHTS = 1.0000000

ROW COLUMN	INDEPENDENT VARIABLES							PREDICTED RESPONSE	STD. DEV. OF PRED. RESPONSE	RESIDUALS	STD. RES.	WEIGHTS
	1 COLUMN	2 COLUMN	3 COLUMN	4 COLUMN	5 COLUMN	6 COLUMN	7 COLUMN					
1	83.00	234300	2356.	60323.000	60055.660	198.63221	267.34060	1.16	1.000			
2	88.50	259400	2325.	61122.000	61216.013	229.14366	-94.013440	-.47	1.000			
3	88.20	258100	3682.	60171.000	60124.712	183.43872	46.287749	.19	1.000			
4	89.50	284600	3351.	61147.000	61597.114	185.99288	-410.11424	-1.70	1.000			
5	96.20	329000	2099.	62221.000	62911.245	239.17173	309.71487	1.64	1.000			
6	98.10	347000	1932.	63639.000	63888.311	185.32858	-249.31092	-1.03	1.000			
7	99.00	365400	1870.	64949.000	65153.048	213.73104	-164.04862	-.75	1.000			
8	100.0	363100	3578.	63761.000	63774.180	216.56573	-13.179974	-.06	1.000			
9	101.2	397500	2904.	66019.000	66004.695	206.11309	14.305143	.06	1.000			
10	104.6	419200	2822.	67857.000	67401.606	175.28850	455.39408	1.83	1.000			
11	108.4	442800	2936.	68169.000	68186.269	182.88234	-17.264911	-.07	1.000			
12	110.8	444500	4681.	66513.000	66552.055	211.49528	-39.054946	-.1A	1.000			
13	112.6	482700	3813.	68655.000	68810.550	186.51199	-155.54985	-.65	1.000			
14	114.2	502600	3931.	69564.000	69649.671	145.68657	-85.671386	-.32	1.000			
15	115.7	518200	4806.	69331.000	68949.069	186.15336	341.93141	1.42	1.000			
16	116.9	554900	4007.	70551.000	70757.758	252.97643	-206.75785	-1.22	1.000			

Figure 5.21b. This first page of the output from the REGRESS module lists only the first three of the independent variables. See the text for a discussion of the remainder of this output. An option is provided to print all of the data vectors entering into this least-squares fit.

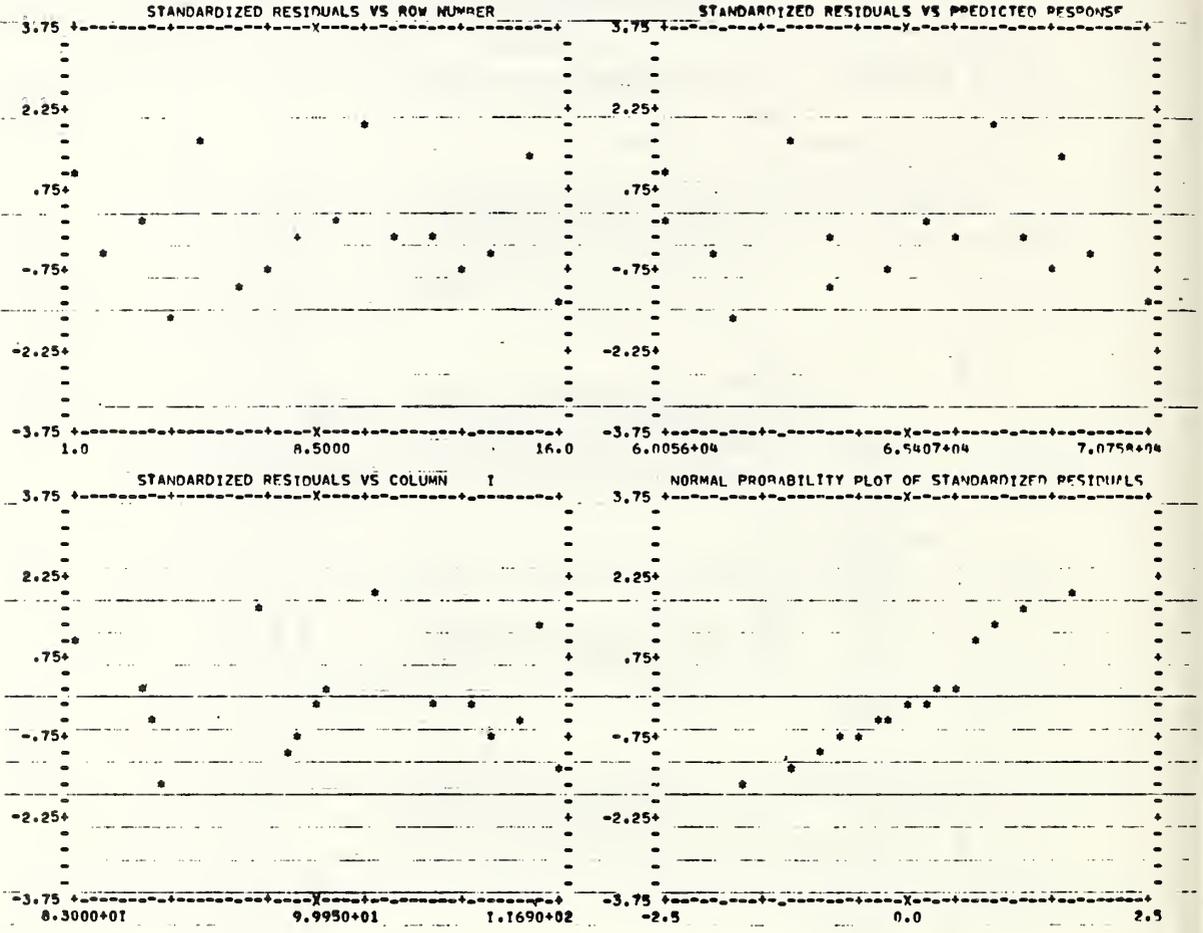


Figure 5.21c. This second page provides a variety of graphical analysis of the residuals (observed values - predicted values).

LEAST SQUARES FIT OF RESPONSE, COLUMN 11,  
 AS A LINEAR FUNCTION OF 7 INDEPENDENT VARIABLES IN COLUMNS 1, 2, 3, 4, 5, 6, 7  
 USING 16 NON-ZERO WEIGHTS = 1.0000000

VARIANCE-COVARIANCE MATRIX OF THE ESTIMATED COEFFICIENTS

COLUMN	1	2	3	4	5	6	7
1	7.2105421+03						
2	-1.8468715	.0011216474					
3	-2.3017172+01	.015467291	.23853409				
4	-6.3467072	.0033628289	.064733739	.045913400			
5	12.654232	-.00633085479	-.083722133	-.0091513252	.051109068		
6	7.2048995+03	-1.2229185+01	-1.8332579+02	-5.3616712+01	39.969388	2.0746056+05	
7	-1.5494989+07	2.43337490+04	3.6355457+05	1.0488363+05	-8.2671279+04	-4.0544122+08	7.9284805+11

ANALYSIS OF VARIANCE  
 -DEPENDENT ON ORDER INDEPENDENT VARIABLES ARE ENTERED, UNLESS VECTORS ARE ORTHOGONAL-

INDEP VAR.	SS=RED. DUE TO COEFF.	CUM. RESIDUAL MS	D.F.	F(COEF=0)	P(F)	F(COEF=0)	P(F)
1	68251199000	12985123.	15	734389.406	.000	105210.889	.000
2	162773470	2285954.8	14	1751.458	.000	347.803	.000
3	5703735.3	2023048.6	13	61.373	.000	67.072	.000
4	6286476.8	1667763.0	12	67.643	.000	68.497	.000
5	16063838.	359028.84	11	172.848	.000	68.781	.000
6	1691494.8	225782.25	10	18.201	.002	16.748	.001
7	1421398.7	92935.979	9	15.294	.004	15.294	.004
RESIDUAL	836423.81		9				
TOTAL	68445977000		16				

Figure 5.21d. These data assist the statistical analyst evaluate the independence of the independent variables and the adequacy of the model.

```

*****
* OMNITAB TEST OF THE REGRESSION MODULE ON THE LONGLEY DATA *
*
*-----*
* LEAST SQUARES FIT OF RESPONSE, COLUMN 11, *
* AS A LINEAR FUNCTION OF 7 INDEPENDENT VARIABLES IN COLUMNS 1, 2, *
* 3, 4, 5, 6, 7 *
*-----*
* USING 16 NON-ZERO WEIGHTS = 1.000000 *
*-----*
* ESTIMATES FROM LEAST SQUARES FIT *
*-----*
* INDEP VAR. COEFFICIENT S.D. OF COEFF. RATIO ACCURACY* *
*-----*
* COLUMN 1 15.061821 .84.914911 .18 4.90 *
* COLUMN 2 -.035819202 .033491005 -1.07 5.64 *
* COLUMN 3 -2.0202296 .48839953 -4.14 6.37 *
* COLUMN 4 -1.0332269 .21427412 -4.82 7.24 *
* COLUMN 5 -.051104037 .22607315 -.23 5.07 *
* COLUMN 6 1829.1514 455.47839 4.02 6.29 *
* COLUMN 7 -3482258.4 890420.16 -3.91 6.28 *
*-----*
* RESIDUAL STANDARD DEVIATION = 304.85403 *
* BASED ON DEGREES OF FREEDOM 16- 7 = 9 *
*-----*
* * THE NUMBER OF CORRECTLY COMPUTED DIGITS IN EACH COEFFICIENT *
* USUALLY DIFFERS BY LESS THAN 1 FROM THE NUMBER GIVEN HERE. *
*-----*
* FIT OMITTING LAST INDEPENDENT VARIABLE *
*-----*
* INDEP VAR. COEFFICIENT S.D. OF COEFF. RATIO *
*-----*
* COLUMN 1 -52.993536 129.54487 -.41 *
* COLUMN 2 .071073197 .030166398 2.36 *
* COLUMN 3 -.42346594 .41773652 -1.01 *
* COLUMN 4 -.57256874 .27899088 -2.05 *
* COLUMN 5 -.41420357 .32128493 -1.29 *
* COLUMN 6 48.417864 17.689486 2.74 *
*-----*
* RESIDUAL STANDARD DEVIATION = 475.16552 *
* BASED ON DEGREES OF FREEDOM 16- 6 = 10 *
*-----*
*****

```

Figure 5.21e. Here we have the computed coefficients and their related accuracies, followed by similar data for a fit in which the last term was omitted. This format of the output was produced at the terminal rather than the line printer on which the other figures were produced.

Figure 5.21f shows how the user interacts with the REGRESS module to produce the results shown in the previous figures. The responses at [A], [B], [C], and [D] transfer to a temporary file, FREGD, the seven data vectors shown in figure 5.21a plus a unit vector (C in column 7). This module next indicates at [E] where the data are stored in the OMNITAB worksheet and provides an opportunity to perform any desired OMNITAB operations. At [F] we choose simply to move the regressand into column 11. At [G] we tell this module which are the independent (regressor) variable, which is the dependent (regressand) variable and the location, if any, of the weights to be assigned to these data.

As we wish to see a plot of the residuals (the observed—calculated values), a positive response at [H] tells us where this module will store the residuals and coefficients.

A response of 'print' at [I] means that we wish all of the output printed. After the negative response at [J], this module prints out the OMNITAB instructions it has generated from the information supplied earlier. As we indicate at [K] that no changes are desired, this module instructs the Exec to carry out the instructions from a file FREGI which contains the above instructions (1-12). The system response at [L] is normal to this type of transfer from Omnidata to OMNITAB.

An examination of figure 5.21e will show that this module produces the coefficients and their standard deviations for a second fit in which the last independent variable is omitted. Since the last named independent variable in column seven was a unit vector, its presence supplies the constant term  $A_0$  in the equation shown above. When this 'variable' is excluded from the fit the resulting equation has no constant term. This feature is parallel to the one in the FIT module where the results are given for a polynomial of degree  $n-1$  in addition to the one of degree  $n$  that was specified.

```

* * * * *
* GOOD AFTERNOON, WELCOME TO OMNIDATA
* * * * *NOTE—OMNIDATA KEEPS A RECORD OF WHO USED WHICH
*      MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
* FILE FTESTREG CONTAINS 8 DATA ITEMS FOR 16 RECORDS
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? >labels,regress (A)
*
* THE FILE FTESTREG CONTAINS DATA LABELED AS FOLLOWS:
* 4  ARMEDF  7  C          2  GNP  1  GNPIPD
* 5  POP     8  TOTALEMP  3  UNEMP 6  YEAR
* *TYPE NAME FOR OMNITAB FILE ---> (B) (C)
* ? >freg
* *TYPE P(ERMANENT) OR T(EMPORARY) FOR FILE --->? >1
* *TYPE A TITLE FOR THIS RUN OR N(ONE)
* ? >test of the regression module on Longley's data
* *TYPE LABELS TO BE TRANSFERRED TO OMNITAB FILE
* (SEQ IS A LEGITIMATE ENTRY) --->
* ? >gnpipd,gnp,unemp,armedf,pop,year,c,totalemp (D)
* *YOUR WORKSHEET IS DIMENSIONED FOR 201 ROWS BY 62 COLUMNS
* *THE DATA ARE STORED IN THE WORKSHEET AS FOLLOWS
* COLUMN LABEL
* 1 GNPIPD
* 2 GNP
* 3 UNEMP
* 4 ARMEDF
* 5 POP
* 6 YEAR
* 7 C
* 8 TOTALEMP
* *INPUT ANY OMNITAB COMMANDS YOU WANT PERFORMED
* BEFORE REGRESS—TYPE END TO END --->
* * * * *

```

Figure 5.21f. Here we see the dialogue between the user and the REGRESS module to carry out a multiple linear regression on some econometric data. See the text for a discussion of the marked section portions.

```

* * * * *
* ? >add 0. to col 8 and store in vector 11 (F)
* ? >end
* *TYPE LABEL OR COL NUMBER OF DEPENDENT Y VARIABLE (REGRESSAND) --->?
* >11
* *TYPE LABEL OR COL NUMBER TO BE USED AS WEIGHTS, OR N(ONE) --->? > none
(G) *TYPE LABELS OR COL NUMBERS FOR INDEPENDENT
    X(I) VARIABLES (REGRESSORS) --->
* ? >1,2,3,4,5,6,7 (H)
* *DO YOU WANT RESIDUALS PLOTTED --->? >yes (I)
* COEFFICIENTS ARE IN COLUMN 72
* RESIDUALS ARE IN COLUMN 61
* *TYPE P(RINT) OR S(TORE) FOR RESULTS --->? >print (J)
* *DO YOU WANT ANY MORE REGRESSIONS --->? >no
* FOLLOWING IS A LIST OF YOUR OMNITAB REGRESS INSTRUCTIONS:
* 1 @ASG,A FREGD
* 2 @USE 7.,FREGD
* 3 @NBS*OMNITAB.
* 4 OMNITAB
* 5 DIMENSION THE WORKSHEET TO HAVE 16 ROWS AND 781 COLUMNS
* 6 FORMAT A (6F12.0/2F12.0)
* 7 READ TAPE A A INTO COLUMNS 1*** 8
* 8 LABEL GNPIPD, GNP, UNEMP, ARMEDF, POP, YEAR, C, TOTALEMP
* 9 ADD 0. TO COL8 AND STORE IN VECTOR 11
* 10 FIT Y IN 11, WTS 1.0 7 IN 1 2 3 4 5 6 7 62 61
* 11 PAGE PLOT COLS 61 vs 1
* 12 STOP
* *DO YOU WANT TO CHANGE ANYTHING --->? >no (K)
*
* TIME : 6.847
* LSD XBASIC 13:53:42 28 SEP 77 (L)
* FACILITY WARNING 100000000000
* READY
*
* OMNITAB TEST OF THE REGRESSION MODULE ON THE LONGLEY DATA
* * * * *

```

Figure 5.21f (concluded).

## 5.22 RENAME

The primary purpose of this module is to change the labels on the data vectors. Thus it is possible to change YOB to YR. OF BIRTH; SS# to SOC.SEC.NO.; PAY to SALARY, etc. This facility is useful for preparing final reports that are easier to read.

If DIV is the name of vector 1, the instruction 1, REGION does the same thing as DIV, REGION, i.e., changes the label DIV to read REGION. If, however, we had entered 1, 500, the label for DIV would remain DIV but its vector number would be changed to 500.

A word of caution: RENAME does not check whether the new name duplicates an existing one. This chore is left to the user. If duplicate names are assigned to data vectors, the system will accept the first one it finds in the label table. Figure 5.22a shows the use of this module to rename a number of data vectors as well as to correct for an inadvertent duplication in labeling the vectors in the DEFINE module.

Normally RENAME retains the original vector numbers unless a request is made to RESEQUENCE. Figure 5.22b shows the result of a resequencing operation on a small data file obtained in the ABRIDGE operation in figure 5.1b. Note that the user must specify the initial value and the interval for the resequencing operation.

\* \* \* N O T E S \* \* \*

```

*****
* FILE FNDEMO CONTAINS 110 DATA ITEMS FOR 500 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? rename
* PRESENTLY. THE LABELS IN THE FILE ARE:
* DIV,SS,SEX,CIT,NAME,DOB,YOB,SCD,STAT,TOA,TOD,TOLA,
* NTE,PP,OSC,FC,TITLE,GRADE,STEP,PAY,PLANT,DEPT,LC,CSOFROM,
* CSDTD,LEGR,DDAY,DDGR,PINFO,DDPP,EUD,EDDY,ADTIT,PROFSTAT
* DEG,COLDEG,YR,SPP,DPG,ATD,SEQ,VP,INS,HC,RET,NOAC,
* SLIM,DLIM,HLIM,EDA,AUTH,POS,AC,PFC,APPOR,REM,AUDAT,
* SUN,SPF,PD,DRPB,CL,NTE,ANU,TP,DET,MC,ALC,SCO,GLUC,
* DAA,RS,SSNC,PF,HBP,N,DPSQ,SP,GDF,SC,DLABR,SFD,HMD,
* WAEPL,WAEDWS,RHWSW,WAEDWP,WAEDWT,LWOP,CNHS,TLWOP,
* CLWOP,CAWOL,CSL,FAB,NDD,WDUD,WED,AA,WP,WWSA,LWUPSA,APAD,
* FUD,DDED,MR,SK,HWWAE,SCDY,SALHAS,C
*
* *TYPE OLD LABEL, NEW LABEL-TYPE END TO STOP-->
* ? CSDFROM,CSDF
* ? PROFSTAT,PROFS
* ? NTE,NTEDA
* ? 63,NTEDP
* ? END
*
* NOW THE LABELS IN THE FILE ARE:
* DIV,SS,SEX,CIT,NAME,DOB,YOB,SCD,STAT,TOA,TOD,TOLA,
* NTEDA,PP,OSC,FC,TITLE,GRADE,STEP,PAY,PLANT,DEPT,LC,
* CSDO,CSDTO,LEGR,DDAY,DDGR,PINFO,DUPP,EUD,EDDY,ADTIT,
* PROFS,DEG,COLDEG,YR,SPP,DPG,ATD,SEQ,VP,INS,HC,RET,
* NOAC,SLIM,DLIM,HLIM,EDA,AUTH,PLS,AC,PFC,APPOR,REM,
* AUDTA,SUN,SPF,PD,DRPB,CL,NTEDP,ANU,TP,DET,MC,ALC,
* SCD,GLOC,DAA,RS,SSNC,PF,HBP,N,DPSQ,SP,GDF,SC,DLABR,
* SED,HND,WAEPL,WAEDWS,RHWSW,WAEDWP,WAEDWT,LWOP,CNHS,TLWOP,
* CLWOP,CAWOL,CSL,FAB,WDD,AUDD,NED,AA,NP,NHSA,LWCPSA,
* APAD,FOD,DDED,MR,SK,RNAME,SCDY,SALBAS,C
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? save
*
* *TYPE NAME UNDER WHICH FILE IS TO BE SAVED -->? fndemo
* FNDEMO IS CATALOGGUED- 64 TRACKS 500 RECORDS 9.1394
*****

```

Figure 5.22a. A record of the use of the RENAME module to correct for duplicate labels and to rename two data vectors. Note: 1) the use of the vector numbers to get at the second NTE name, and 2) response of the operation to SAVE the file with the revised label table. RENAME always displays the list of current labels before and after the operations are performed.

```

* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? labels
*
* THE FILE FSFCP CONTAINS DATA LABELED AS FOLLOWS:
* 11 AMOUNT 16 AM72 2 CCODE 3 CIST 8 COUNTRY
* 13 DB 15 DE 10 DIV 1 ID 4 PROGC
* 12 YB 14 YE
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? rename
*
* PRESENTLY, THE LABELS IN THE FILE ARE:
* ID, CCODE, CINST, PROGC, COUNTRY, DIV, AMOUNT, YB, DB, YE, DE,
* AM72
* *TYPE OLD LABEL , NEW LABEL -TYPE END TO STOP --->
* ? amount,cost
* ? resequence
* *ENTER INITIAL VALUE AND STEP --->? 1.1
* ? end
* NOW THE LABELS IN THE FILE ARE:
* ID, CCODE, CINST, PROGC, COUNTRY, DIV, COST, YB, DB, YE, DE,
* AM72
*
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? labels
* THE FILE FSFCP CONTAINS DATA LABELED AS FOLLOWS:
* 12 AM72 2 CCODE 3 CINST 7 CUST 5 COUNTRY
* 9 DB 11 DE 6 DIV 1 ID 4 PROGC
* 8 YB 10 YE
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? stop
* PROGRAM STOPPED.
* * * * *

```

Figure 5.22b. An application of the RENAME module to change certain labels and to resequence the labels, showing: 1) the condition of the file after abridging, especially the non-sequential vector numbers; 2) the renaming of one data vector; 3) the instruction to resequence; and 4) the resequenced labels.

### 5.23 The REPORT Module (Concluded)

In this section we continue the discussion started in section 3.3 by explaining additional features which are activated via responses alternate to those given earlier. The more important of these provide for:

- a) inserting literal expressions between columns of a report;
- b) indicating labels to be excluded from a report when the list of labels is much longer than the list of exceptions;
- c) suppressing the paging of a report;
- d) options to write the report on a file in addition to or instead of the terminal; and,
- e) interrupting the printing.

In the examples shown in section 3.3 we have seen how it is possible to shift column headings around. Similarly it is useful to be able to insert blanks or other arbitrary character strings between the columns in a report. If we wished to insert a colon between the data fields shown in figure 3.3b, it is simply necessary to type as follows:

```
SS#,*,*,DOB,*,*,NAME
```

instead of simply

```
SS#,DOB,NAME.
```

In some data bases, fields are defined wider than is necessary and the report module allows space in accord with the defined field width rather than the space taken up by the actual entries. This situation can be seen in the name field in figures 3.3a and 3.3b as well as in the ID NO. field in figure 3.3c. It is possible to take out the extra spaces following any item of data by following the label with SHORT, n. Here n is the number of characters to be allocated to that column in the report. If we wished to reduce the space between the first two columns in the report shown in figure 3.3c, the instructions for LINE 1? would be ID,SHORT,11,DIV,YB,YE,CCODE,CINST,AMOUNT.

When the report module asks the user to:

TYPE LINES FOR OUTPUT,

the response can be:

- a) a list of labels;
- b) a mixture of labels and interspersed arbitrary character strings (embedded between asterisks);
- c) ALL (meaning all data items); and,
- d) ALL,EXCEPT,A,B,C where A,B, and C are proper labels.

The REPORT module has been programmed to respond to the normal UNIVAC interrupt feature without returning control back to the operating system (EXEC). If it becomes necessary to interrupt a report while it is printing on-line, the user should type @@X TIOC and return the carriage. On the next line, if we type STOP the Omnidata run is stopped. In order to stay in the REPORT module it is necessary to type RUN and return the carriage. At this stage the REPORT module has been alerted by the EXEC that an interrupt has occurred and allows the user to restart the report again, just as if he had used the RESTART option earlier.

```

* * * * *
* *TYPE MODE--L(ABELED) OR UNL(ABELED) --->? >I *
* *TYPE CHARACTERS PER LINE --->? >80 *
* *TYPE LINES FOR OUTPUT *
* LINE 1 ? >all (A) *
* LINE 26 ? >end *
* *TYPE TITLE LINES OR N(ONE) ---> *
* ? >n *
* *TYPE FOOTNOTE LINES OR N(ONE) ---> *
* ? >n *
* PLEASE POSITION PAPER *
* ? > *
* ----- *
* INST= 3 SS#= 410064386 SEX= 1 CIT= 1 *
* NAME= JUJIGUQE, MILA A. [MR. ] DOB= 101928 YOB= 28 *
* SCD= 061662 STAT= 1 TOA= 1 TOD= 1 TOLA= 2222 NTE= *
* PP= GS USC= 08020 FC= *
* TITLE= ENGINEERING TECHNICIAN *
* *
* GRADE= 12 STEP= 06 PAY= 19462 DIV= 314 SEC= 31402 *
* LC= 240630031 CSDFROM= 061266 CSDTC= 061266 LEGR= 51 *
* DDAY= 06117 DOGR= 061266 PINFO= 121001 DOPP= 061668 EOD= 061266 *
* EODY= 66 ADFIT= 00 PROFSTAT= 2 DEG= 0 COLDEG= 000000 YR= 00 *
* SPP= DPG= ATD= SEQ= 4100643866 VP= 22 INS= 1 *
* HC= RET= 1 NOAC= 89400 SLIM= DLIM= HLIM= *
* APPOR= 00 REM= 081153 AUDAT= 061172 SON= *
* SPF= 22222 PD= DRPB= CL= 145 NTE= *
* ANU= 2 TP= 2 DET= 2 *
* MC= 7 ALC= 5 SCD= GLOC= DAA= RS= 00 *
* SSNC= 019205318 PF= 1 HBPN= 102 DPSO= SP= *
* GDF= SC= DLABR= SED= 000000 HWD= 0000 *
* WAEPL= 0000 WAEDWS= 000 RHWSW= 0000 WAEDWP= 000 WAEWT= 0000 *
* LWOP= 0000 CNHS= 0000 TLWOP= 00 CLWOP= 0000 CAWOL= 0000 *
* CSL= 0038 FAB= 2221222222220000000010 WDD= 000000 WDUD= 060974 *
* WED= 000000 AA= 000 NP= 00000 NHTA= 0000 LWOPSA= 000 *
* APAD= 00000000 FUD= 000000 DDED= 000000 WH= 0 SK= 0 *
* HWWAE= 0000 SCDY= 62 SALBAS= PA C= 00 *
* * * * *

```

Figure 5.23a. Here we see a more compact printout (of all of the data items for one record) than is provided by the DISPLAY module (see fig. 5.10). Note at [A] that this module determines that 25 lines will be required to display all of the requested information and then asks what is desired on line 26.

```

* * * * *
*   *TYPE MODE--L(ABELED) OR UNL(ABELED) --->? >unl | *
*   *TYPE DIMENSIONS OF LINE AND PAGE --->? >80 *
*   *TYPE LINES FOR OUTPUT *
* LINE 1 ? >rl,dm,formula,short,30 *
* LINE 2 ? >end *
*   *TYPE TITLE LINES OR N(ONE) ---> *
* ? >none *
*   *ANY COL HEADS--TYPE L(ABELS)<N(ONE) OR 3 HEADINGS ---> *
* ? >labels *
* ? >end *
*   R1           DM           FORMULA *
* 0.0903        1.403        C17.H35 C O O AG *
*   *OK--TYPE YES OR NO --->? >yes *
*   *TYPE FOOTNOTE LINES OR N(ONE) ---> *
* ? >n *
*   *DO YOU WANT TO POSITION PAPER AFTER EACH PAGE-- *
*   TYPE YES OR NO --->? >no *
* ----- *
*   R1           DM           FORMULA *
* == == == == == *
* 0.0963        1.403        C17 H35 C O O AG *
* 0.1065        1.452        C15 H31 C O O AG *
* 0.1090        0.95         C H3( C H2)7 C H= C H( C H2) *
* 0.1168        0.95         C H3( C H2)7 C H= C H( C H2) *
* 0.1192        1.506        C13 H27 C O O AG *
* 0.1263        0.97         C19 H36 O2 *
* * * * *

```

Figure 5.23b. In this report from a file containing crystal data, we see the utility of the SHORT, 30 modifications in the formula field.

## 5.24 SAVE

This module allows the user to store away on mass storage and catalogue the *current* file for use in a later run. The name under which the file is to be saved is typed in response to the request:

TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->?

If the file name duplicates an existing file in the user's catalogue, the module responds with:

DUPLICATE FILE NAME. TYPE A NEW NAME OR R(EPLACE)-->

Figures 5.24a and 5.24b show typical uses of this module.

One might well wonder why this module asks for another input at [B]. There are two reasons. If instead of END we had typed any other string of characters excluding AUTO (the use of which is described in the next paragraph), this module would have saved away another copy of the file under that name. Second, and more important, it is possible to delete files by responding with the word DELETE instead of END. The name of the file to be deleted is supplied subsequently in response to:

TYPE FILE TO BE DELETED--->?

This feature allows one to delete catalogued files without becoming proficient in using the EXEC 8 commands.

If the user responds with the word AUTO instead of a file name, this module names the file automatically as seen at [B] in figure 5.24b to consist of the letters PF followed by 9 or 10 digits. The digits represent in order: the month, the day of the month, the hour, the minute, and the second at which the file was saved. This feature is useful in keeping track of files that are updated or otherwise modified frequently.

\* \* \* N O T E S \* \* \*

```

* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* --->? >save *
* * * * *
* *TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->? >fndemo *
* FNDEMO IS CATALOGUED—64 TRACKS 500 RECORDS 16.4226 *
* *TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->? >end*
* CPU SEC IN SAVE = 2.0888 *
* CPU SEC = 17.8322 TIM = 16:22:53 *
* * * * *

```

Figure 5.24a. Here is a dialogue with the SAVE module to save a current file under the name FNDEMO as at [A]. The user has an opportunity at [B] to save the file again under another name or to delete an earlier file as is indicated in the text.

```

* * * * *
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS *
* *TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->? auto *
* PF6100163952 IS CATALOGUED—64 TRACKS 500 RECORDS 16.6406*
* *TYPE NAME UNDER WHICH FILE IS TO BE SAVED --->? end *
* * * * *
* CPU SEC IN SAVE = 1.8668 *
* CPU SEC = 17.7914 TIM = 16:43:55 *
* * * * *

```

Figure 5.24b. Results from the use of the automatic naming feature at [A] of the SAVE module. See the text for an explanation of the numbers in the file name at [B].

## 5.25 The SEARCH Module (Continued)

In this section we continue the discussion started in section 3.1 by explaining the more specialized features of this module. The more important of these provide for:

- a) ignoring specific characters in the file during the searching operation;
- b) locating only the first record or first contiguous block of records that satisfies a given criterion;
- c) limiting the search to a specific block of records in the file;
- d) selecting records in which the entry (value) for one vector (attribute) equals that for another vector;
- e) locating records having missing (blank) data entries in certain data vectors; and,
- f) taking into account the order and distance between string fragments or words in a data item.

We now take up these features in turn.

When the SEARCH module is called it requires the user to respond to the following:

\*TYPE S(ELECT) OR R(EJECT) --->?

If instead of the normal response discussed in section 3.1, the user types the word IGNORE, the module is prepared to accept a sequence of characters which it will ignore in making its comparison in subsequent searches. This feature was implemented to facilitate searching fields which contain such extraneous characters as shift symbols for upper and lower case or control character sequences. It has more general utility for ignoring hyphens or punctuation marks, parentheses, etc.

If we respond to the request:

\*TYPE LABEL AND VALUE --->?

with

FIRST, REACTANT, COH OR CN

the search module will extract from the file only the first record or the first block of records in which COH or CN appear in the reactant vector. Other special responses to the above question are also accepted. Thus:

LIMIT,1,1000

restricts the search to the first 1000 records in the file, while LIMIT,n,m confines the search to the block of records from the nth to the mth inclusive.

A response of

LABEL,L:LABEL

will select from a file all records in which the value found in the first named label is equal to that found in the second. When applied to a

federal personnel file, the request EOD,L!SCD finds all persons whose service computation data (SCD) coincides with the date they entered on duty (EOD); thereby locating persons who had no government employment prior to joining their present agency.

The exclamation point following the L in the above instruction serves to make it clear that the characters SCD refer to a legitimate label and not to the value of the label EOD. The exclamation point has another use as follows.

MOLECULE,C!N

will locate C2, C12, C4F8 but not C, CH, CN, etc. If the exclamation point is followed by the letter A as in MOLECULE, CO!A, the SEARCH will locate CON, COB, COC, etc. but not CO2. Additionally

MOLECULE,CO!G

will locate CO., CO+, CO-, CO:, etc. It should be clear from the above that the letters !N, !A, and !G are general representations for any numeric, any alphabetic, or any graphic respectively. This feature is important in handling chemical formulas to be able to select carbon (C) or (C2) without getting calcium (CA), cobalt (CO), etc.

It is often useful and sometimes necessary to determine which records in a file have missing data items (blank entries). The SEARCH module facilitates this by accepting the word BLANK in a search instruction (GRADE,BLANK) to mean that the entire entry in the GRADE field should be blank.

In the search options discussed thus far nothing was said concerning the ordering of the strings in the file. Thus if we responded to the search module by typing

TITLE, PHYS AND CHEM,

persons whose job titles were PHYSICAL CHEMIST or CHEMICAL PHYSICIST would satisfy the search equally. If we wished the SEARCH module to take the word order into account we would type:

TITLE, \*SEQ\*, PHYS AND CHEM

in which case chemical physicists would not be selected.

A more meaningful example is afforded by a file containing chemical reactions such as are shown in figure 5.25a where the entire reaction is contained in a single data vector called REACTION. In searching this file it is necessary to differentiate between reactants to the left of the arrow and products to the right of it. If we wished to retrieve all reactions in which H2 is a reactant and O2 is a product, the instruction

REACTION,\*SEQ\*,H2,---> ' O2 '

would retrieve reaction 112 but not reaction 11. In contrast sequence REACTION, H2 AND O2 would yield both reactions 11 and 112 plus all others where the reactants are H<sub>2</sub>O<sub>2</sub>.

It should be observed that when we ask for a sequential search such as:

TITLE,\*SEQ\*,A,B,C

where A, B, and C are arbitrary character strings we imply variable length open-ended ellipses. Thus, any character string such as ..A...B....C.... would satisfy the search. In many practical instances it is important to restrict the distance between the strings A, B, and C. A provision has been made for this proximity feature by allowing the user to specify a maximum distance (either in characters or words) between the specified strings. This is done as follows:

TEXT,\*SEQ\*,A,..nC,B,..nC,C etc.

or

TEXT,\*SEQ\*,A,..5W,B,..3W,C

where A, B, and C are specific strings; ..nC denotes that the strings should be at most n characters apart. If instead of ..nC we had written ..nW we would require the specified strings to be at most n words apart. It is also possible to mix character distances and word distances in a single instruction.

This feature would enable one to greatly streamline the search instructions when searching in a data file on organic compounds for molecules having less than 10 hydrogen (H) or 10 carbon (C). atoms. Thus:

FORMULA,\*SEQ\*,C,..1C,H,..1C,!A

would locate C6H6,CH4,C4H9N3O, etc. but not C8H14O or C12H9BR.

The ability to specify that the two designated character strings in a search instruction be closely coupled is useful when searching a lengthy abstract. Thus

ABSTRACT,\*SEQ\*,AGRICULTURAL,..1W,PRICE

would serve to retrieve abstracts containing the words: agricultural prices, agricultural price index, agricultural commodity prices. Text in which these words were farther apart would not be retrieved.

Finally, there is another variant of the search instruction which is useful in coping with files in which the dates are entered in nonstandard format (062575 for June 25, 1975) instead of the standard form (750625). Thus the command in SEARCH:

DOB,\*SDF\*,400123

will locate all persons born on Jan. 23, 1940 even though the date is entered in the file as 012340. In the above, \*SDF\* alerts the search module that the date entered in the file is not in the standard format specified in the search request, but rather must be converted to the standard data format of year, month, day (YYMMDD).

* NO.	REACTION	AUTHOR YEAR	REMARK
* 11	H + O2 + H2 --> HO2 + H2	BASCOMBE	1965 EVAL
* 18	HO2 + H2 --> HO + H2O	JENSEN	1967 REC
* 20	H2 + O --> H + HO	SCHOFIELD	1967 EVAL
* 21	HO + H2 --> H + H2O	SCHOFIELD	1967 EVAL
* 47	H2 + O --> H + HO	BAULCH	1968 EVAL
* 49	HO + H2 --> H + H2O	BAULCH	1968 EVAL
* 61	HO2 + H2 --> H + H2O2	BAULCH	1969 EVAL
* 70	CH3 + H2 --> CH4 + H	WALKER	1968 EVAL
* 107	HO + H2 --> H + H2O	WILSON	1972 EVAL
* 111	HO2 + H2 --> H + H2O2	LLOYD	1970 EVAL
* 112	HO2 + H2 --> H + O2 + H2	LLOYD	1970 REV
* 165	H2 + O --> H + HO	KONDRATIFV	1970 REV
* 166	H2 + O --> H + HO	BAULCH	1968 EVAL

Figure 5.25a. Here we see five data elements in a chemical reaction file where the sequential feature of the SEARCH module has an important application as discussed above.

```

* * * * *
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >search
*   *TYPE S(ELECT) OR R(EJECT) --->? >s
*   *TYPE CHARACTERS TO BE IGNORED, OR NONE --->? >none
*   *TYPE LABEL AND VALUE(S)
* 1 --->? >eod,05,09
*   AND
* 2 --->? >end
* 2 HIT(S) WHEN:
*   EOD IS 05 AND
* 09
*   *TYPE D(ISPLAY),S(EARCH) OR E(XIT) --->? >p,c
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >eod
*
* EOD
* 090568
* 050996
*   *TYPE D(ISPLAY),S(EARCH) OR E(XIT) --->? >s,o,s
*   *TYPE CHARACTERS TO BE IGNORED, OR NONE --->? >none
*   *TYPE LABEL AND VALUE(S)
* 1 --->? >eod,*seq*,05,09
*   AND
* 2 --->? >end
* 1 HIT(S) WHEN:
*   EOD IS IN SEQUENCE 05 AND
* 09
*   *TYPE D(ISPLAY),S(EARCH) OR E(XIT) --->? >p,c
*   *TYPE LABELS OF ITEMS TO BE DISPLAYED OR ALL
* --->? >eod
*
* EOD
* 050966
*   *TYPE D(ISPLAY),S(EARCH) OR E(XIT) --->? >s,o,s
*   *TYPE CHARACTERS TO BE IGNORED, OR NONE --->? >none
*
* * * * *

```

Figure 5.25b. A record of a run to test the sequential search feature of this module. Note how the response P,C at [A] prints out the information from the current file. The response S,O,S at [B] indicates that we want to SEARCH on the ORIGINAL file in the SELECT mode. The system recognizes this multiple answer and skips over the corresponding questions.

## 5.26 SEGMENT

The practical implementation of computer subroutines in general, and Omnidata modules in particular, often requires the establishment of upper limits on the sizes of data arrays that can be handled. When the size of an Omnidata file exceeds these limits, it is often possible to carry out the desired operations on portions of the file. Facilities for partitioning files exist in the ABRIDGE, SEARCH, and DISTRIBUTE modules discussed earlier.

The SEGMENT module allows for partitioning of Omnidata files in a number of ways—all different from the ways available in ABRIDGE, SEARCH, and DISTRIBUTE. This module can segment a file into blocks of contiguous records in the following ways:

- a) in multiples of N logical records;
- b) at record numbers a, b, c, etc.

The facility for segmenting a file into uniform blocks of records is useful in streamlining subsequent operations, and requires no detailed knowledge of the distribution of the records. A segmentation of the file at specific discrete records (a, b, c, etc.), on the other hand, does imply a knowledge of the unique character of the records a, b, c, etc. Such information would normally result from an examination of a complete or partial listing of the file—a condition which is not at all unusual.

A listing of any bibliographic data file arranged alphabetically by the first author will show a nonuniform distribution of authors among the letters A–Z. In that case it may be useful to break the file so that names starting with the letter A–D, E–J, K–P, are grouped together while names starting with the letter S comprise a separate file. If a listing exists from which one can learn the record numbers associated with the starting positions of the names beginning with A, E, K, S, etc., it will be much more efficient to use this module than it would be to use the DISTRIBUTE module which has to read each record to determine to which file it should be sent.

In figure 5.26a we see how this module accepts instructions and reports the result of segmenting a bibliographic data file into five files covering the years: prior to 1960, 1961–1970, 1971–1975, 1976, 1977. A previous listing of the file in chronological order yielded the following record numbers: 28, 273, 771, and 868, representing the ending points of each block except the last.

If the response at [A] had been a single number, say 500, the file would be divided into two pieces. The first piece would contain the first 500 records, and the second would contain the remainder of the file. If the response had been 500\*, this module would have segmented the file into blocks of 500 records each.

```

* * * * *
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* -->? >segment
*   *TYPE ONE OR MORE RECORD NUMBERS
*   *AT WHICH THE FILE IS TO BE SEGMENTED
* -->? >28,273,771,868
*
*   YOU HAVE CREATED 5 FILES
*   NAME           # OF RECORDS
* FSEG1           28
* FSEG2           245
* FSEG3           498
* FSEG4           97
* FSEG5           8
* * * * *

```

Figure 5.26a. A record of the dialogue with the SEGMENT module to break a file at the record numbers indicated at [A]. See fig. 5.11b for an alternate way of breaking the file via DISTRIBUTE when the information required at [A] is not available. See the text for an explanation of how this module interprets a response of 500\* at [A].

\* \* \* N O T E S \* \* \*

## 5.27 SEQUENCE

This module can perform two functions. The first is to add a sequence number to an existing file for each logical record starting with any designated integer and proceeding by a designated interval. It can also tag each record with a fixed character string, either numeric or alphabetic. The vector or vectors added are named SEQ #1 and FLAG 1 and are added to the *label table* and to the *label file*. When called, this module reads the *label table* to see if the file had already been sequenced or flagged. If it has, the module offers the option of writing over the existing flags or sequence numbers or of generating a new vector of flags or sequence numbers or both. Figure 5.27a shows a typical operation of this module. The REPORT module has a provision for outputting sequence numbers, but those are not made part of the file.

The provision for resequencing or reflagging a flagged or sequenced file is useful when the current version of the file represents a portion or a modification of the file that was originally sequenced or flagged. Different cuts can thus be flagged for future reference. In order that these different cuts remain consonant for future stacking of the files, we have allotted 5 characters for each new sequence field or flag field created. If the user requests an initial value or a step which results in a sequence number greater than 99999 he is given an appropriate diagnostic and asked to reenter values for the starting number and the step. If the user enters more than five characters for the flag, it is truncated and an appropriate diagnostic is given.

```
* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >fpr75
* FILE FPR75 CONTAINS 111 DATA ITEMS FOR 75 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >sequence
*
*   *TYPE INITIAL VALUE AND STEP --->? >1,1
*   *TYPE FLAG IF ANY, OR N(ONE) --->? >nbs
* THE SEQUENCING IS IN VECTOR SEQ#1
* THE FLAG IS IN VECTOR FLAG1
*
* CPU SEC IN SEQUENCE = 4.2282
* CPU SEC = 6.8116 TIM = 0:54:12
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >sequence
*
*   *TYPE INITIAL VALUE AND STEP --->? >100,10
*   *TYPE FLAG IF ANY, OR N(ONE) --->? >none
* SEQ#1 ALREADY EXIST(S)
*   *TYPE "new", OR LABEL TO BE REPLACED --->? >new
* THE SEQUENCING IS IN VECTOR SEQ#2
* * * * *
```

Figure 5.27. Two consecutive uses of the SEQUENCE module on the same file showing: 1) the instruction to sequence the records starting from 1; 2) the manner of inserting a constant flag vector; 3) the names that are automatically assigned to the sequence vector and to the flag vector. On the second or subsequent attempts to sequence or flag this file, we are told at 4) that a vector SEQ #1 already exists; provided an opportunity at 5) to write over or generate a new vector; and told at 6) that a new vector SEQ #2 contains the new sequence numbers.

## 5.28 SORT

This module allows the user to sort a file on as many as 10 data vectors in any desired order, as long as the total length of the resulting *sort key* is 60 characters or less.

The user is first asked to

\*TYPE ALPHA(BETIC) OR VAL(UE) SORT --->

An answer of ALPHA or ALPHABETIC will sort the file based on the fielddata equivalents of the characters in the data vectors being sorted. The response VAL or VALUE instructs the module to use numeric values, and is necessary to prevent such sequences as 1., 10, 2., etc. which would occur if the number were left adjusted in the designated field. If the numbers are entered with the decimal points aligned, the alphabetic sort will be proper and more efficient.

After this, the order in which the file is to be sorted is determined based on the order of the labels supplied in response to the request:

\*TYPE SORT KEYS IN ORDER--->? >

Since sorting is often a prerequisite for preparing summarizations for budgetary and other administrative purposes, the SORT module performs certain arithmetic operations during the building of the sorted file. Figure 5.28a shows a typical use of the arithmetic facility in the SORT module, where we compute and display the number of items and certain averages, subtotals, totals, and grand totals of the data found in the three data vectors: YOB, EODY, and PAY. Note that averages, totals and grand totals, and grand averages are struck at each of the designated levels indicated in the third instruction. Here we have struck totals on the same vectors as were sorted. The response to the third question must be one or more of the vectors on which the file was sorted.

At present the SORT module can handle only 3550 records. If the file to be sorted exceeds this number, the user is instructed to break the file into two or more manageable portions via the DISTRIBUTE module operating on the data vector which serves as the leading portion of the *sort key*. These files can then be sorted in turn and then stacked using the STACK module.

If the response to the second question were N(ONE), this module would simply have sorted the file in accord with the first instruction and reminded the user to SAVE it. Since sorting is a relatively expensive operation on a large file, it is advisable to keep the file in sorted order until such time as it may be necessary to resort to a different order.

```

*****
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS                               *
*   ->? sort                                                                *
*   *TYPE SORT KEYS IN ORDER -->?region,plant,sex                         *
*   *TYPE LABELS TO BE TOTALLED AND AVERAGED -->? yob,eody,pay          *
*   *TYPE LABELS AT WHICH TO STRIKE GRAND TOTALS AND AVERAGES-- ->      *
*   ? region,plant,sex                                                    *
*   FLAGS OBTAINED-TIM = 22.278 CLK = 142825                               *
*   REGION    PLANT    SEX    LABEL    TOTAL    AVERAGE
*   1         00      1      YOB      4        124        31
*   1         00      1      EODY     4        261        65.25
*   1         00      1      PAY      4        115172    28793
*   1         00      2      YOB      4        145        48.3333
*   1         00      2      EODY     3        212        70.6667
*   1         00      2      PAY      3        24702    8234
*   1         00      TOTAL  YOB      7        269        38.4286
*   1         00      TOTAL  EODY     7        473        67.5714
*   1         00      TOTAL  PAY      7        139874   19982
*   1         29      TOTAL  YOB     10       360        36
*   1         29      TOTAL  EODY    10       661       66.10
*   1         29      TOTAL  PAY     10      34415     3441.5
*   1         TOTAL  YOB    119      3879     32.5966
*   1         TOTAL  EODY   119      7959     66.8824
*   1         TOTAL  PAY    119     859800   7225.21
*   2         00      1      YOB      1        21         21
*   2         00      1      EODY     1         48         48
*   2         00      1      PAY      1      15331    15331
*   6         40      2      YOB      1         26         26
*   6         40      2      EODY     1         67         67
*   6         40      2      PAY      1      15860    15860
*   6         40      TOTAL  YOB      2         79         39.5
*   6         40      TOTAL  EODY     2        140         70
*   6         40      TOTAL  PAY      2      22742    11371
*   6         TOTAL  YOB    22         666        30.2727
*   6         TOTAL  EODY   22        1437        65.3182
*   6         TOTAL  PAY    22      426930    19405.9
*   TOTAL     TOTAL  YOB   336      10931     32.5327
*   TOTAL     TOTAL  EODY  336     21549     64.1339
*   TOTAL     TOTAL  PAY   336    4.796171E+6  14274.3
*   CURRENT FILE IS SORTED-CALL SAVE TO SAVE.
*****

```

Figure 5.28a. Instructions for and output from a SORT operation on a personnel file showing: 1) the sort keys in order desired (REGION, PLANT, and SEX); 2) the data vectors upon which the built in arithmetic is to be performed; 3) the hierarchical levels at which the subtotals and subgroup averages should be struck and printed; and finally, 4) a reminder to SAVE the file if necessary.

```

*****
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? sort
*   *TYPE SORT KEYS IN ORDER -->? region,plant,sex
*   *TYPE LABELS TO BE TOTALLED AND AVERAGED--->? yob,cody,pay
*   *TYPE LABELS AT WHICH TO STRIKE GRAND TOTALS AND AVERAGES -->
*   ? region,plant
*   FLAGS OBTAINED--TIM = 43.279 CLK = 144127
*   REGION    PLANT    LABEL          TOTAL    AVERAGE
*   1         00      YOB            7      ● 269      38.4286
*   *         *      EODY           7      ● 473      67.5714
*   *         *      PAY            7      ● 139874   19982
*   1         10      YOB            2        53       26.5
*   *         *      EODY           2       106       53
*   *         *      PAY            2     57671    28835.5
*   1         15      YOB            1        27       27
*   *         *      EODY           1        69       69
*   *         *      PAY            1    31383    31383
*   1         21      YOB            8       256       32
*   1         29      YOB           10       360       36
*   *         *      EODY          10       661      66.10
*   *         *      PAY           10     34415    3441.5
*   1         TOTAL  YOB           119     3879     32.5966
*   *         *      EODY          119     7959     66.8824
*   *         *      PAY           119    859800   7225.21
*   2         00      YOB            3        78       26
*   *         *      EODY           3       170     56.6667
*   *         *      PAY            3     48130   16043.3
*   2         05      YOB            5       153       30.6
*   *         *      EODY           5       294       58.8
*   *         *      PAY            5    122288   24457.6
*   6         40      YOB            2        79       39.5
*   *         *      EODY           2       140       70
*   *         *      PAY            2    22742   11371
*   6         TOTAL  YOB            22       666     30.2727
*   *         *      EODY          22     1437     65.3181
*   *         *      PAY            22    426930  19405.9
*   TOTAL    TOTAL  YOB            336     10931    32.5327
*   *         *      EODY          336     21549    64.1339
*   *         *      PAY            336    4.796171E+6 14274.3
*   CURRENT FILE IS SORTED -- CALL SAVE TO SAVE.
*****

```

Figure 5.28b. Here we see a less detailed summary file than was shown in the previous figure. Note that the totals tabulated here correspond to the TOTALS struck in the earlier figure.

## 5.29 STACK

The STACK module combines consonant files into a single composite file whenever the user feels it would be advantageous or necessary to have the larger file for subsequent Omnidata operations. For example, if one had 50 separate files, each containing data for a single state, and wished to analyze the data for a particular area of the country, he could STACK the files for the states in that area into a single file, and then perform the analysis once on the larger file. Please note that by consonant files we mean that the files should have the same record length and same record layout for the data elements. The names (labels) for the data elements need not be identical. Thus the label in one file can be YR, and in another it can be YEAR. The labels for the composite file are taken from the *first* of the stacked files.

The user is first asked to

\*TYPE FILES TO BE STACKED  
(NOTE: ONE IN CORE ASSUMED FIRST.)--->

This indicates that the first file to be copied to the composite file is the one in core, the one the user entered the Omnidata system with, or a derivative of it, if operations (searching, sorting, etc.) have been performed on it. After this, the files are stacked in the order in which they are named in response to the above question.

If any named file cannot be assigned, the user is given an error message as to why, and asked to

\*TYPE CORRECT FILE NAME, SKIP, OR END --->

Appropriate action is taken depending on the response. Similarly, if the record length of any file is found to be different from that of the first file, an error message is printed and the user is asked whether he wants to skip the file at fault, or end the stacking operation.

After all files have been successfully assigned and have been found to have consonant formats, the user is requested to

\*TYPE NAME FOR NEW COMBINED FILE --->

The actual stacking is then accomplished, and the user then has a permanently catalogued file with the name given above. It is important to note that after the STACK, when control is returned to Omnidata, the user has in core the same file he entered STACK with, *not* the newly saved composite file. Should he want this composite file, he can get it via the FETCH module.

```

* * * * *
* *PLEASE ENTER ACCOUNT NUMBER --->? >xxxxx (A)
* *TYPE PASSWORD --->? >xxx
* *WHICH DATA BASE DO YOU WANT --->? >anor-org
*
* GOOD MORNING, WELCOME TO OMNIDATA
* * * *NOTE-OMNIDATA KEEPS A RECORD OF WHO USED WHICH * * *
* * *MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*
* FILE ANOR-ORG CONTAINS 18 DATA ITEMS FOR 668 RECORDS
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stack,monitor,250
* *TYPE FILES TO BE STACKED. (B)
* (NOTE:ONE IN CORE ASSUMED FIRST.) --->? >
* ? >anor-inorg (D)
(C) *TYPE NAME FOR NEW COMBINED FILE --->? >anorthic
250 RECORDS COPIED - 2.5116
* 500 RECORDS COPIED - 2.782
* FILE ANOR-ORG COPIED 2.9548
* 750 RECORDS COPIED - 3.0378
* 1000 RECORDS COPIED - 3.2966
* FILE ANOR-INORG COPIED 3.3246 (E)
*
* CPU SEC IN STACK = 2.9642
* CPU SEC = 4.6354 TIM = 10:26:25
(F) *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >length
* PRESENTLY, YOUR FILE HAS 1026 RECORDS.
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >stop
* PROGRAM STOPPED.
* * * * *

```

Figure 5.29a. Here we have a dialogue with the STACK module. As we entered Omnidata at [A] with the file ANOR-ORG, this module reminds us of this fact at [B] when requesting a list of files to be stacked. At [C] we ask that the inorganic file (ANOR-INORG) be stacked under the organic file. The name for the composite file is entered at [D] and at [E] we see how the MONITOR reports progress in the operation. If the MONITOR has not been set, the exact length of the file can be obtained by typing LENGTH when the system asks for a module or an instruction at [F].

## 5.30 STATIS

Omndata is unique among data management systems in regard to the depth and the ease of performing statistical analysis. In particular, the STATIS module, which is patterned after the STATIS command of the original OMNITAB system developed by Hilsenrath et. al., prints out for any single numeric data vector in the file the following:

- a. a histogram and frequency distribution in deciles of the data entries including cumulative frequency and percent and cumulative percent;
- b. three measures of location: the weighted and unweighted mean and the median;
- c. six measures of dispersion;
- d. six tests for nonrandomness;
- e. nine computations on the deviations; and,
- f. nine other statistics including Student's T, measures of kurtosis, etc.

After the above items are printed, an option is provided for printing out an extensive auxiliary table consisting of: the entire data vector in its natural order; the deviations from the mean; the rank of each data point; the data rearranged in increasing order; and the difference between successive values of the ordered data items.

Figure 5.30a shows the result of a statistical analysis of the ages of 429 persons in the test file FNDEMO429. It is beyond the scope of this handbook to discuss the meaning of the statistical parameters produced by this module. The reader is referred to NBS Technical Note 756 by H.H. Ku entitled A Users' Guide to the OMNITAB Command "STATISTICAL ANALYSIS" (March 1973) for an explanation of these results.

At [A] in figure 5.30a we see a frequency distribution of the data in deciles. If such an analysis is desired showing midranges in the decile column, it can be achieved by suitably encoding in the ENCODE module (see sec. 5.12). Had we responded at [B] with YES we would get 429 lines of output of the type shown in figure 5.30b.

The statistical analysis carried out by this module can be performed only on numeric data vectors. It becomes important, therefore, to decide what action this module will take when it encounters a nonnumeric data point (a mistake)—a situation which, unhappily, occurs too often. STATIS does not come to a screeching halt when it encounters a few mistakes. It discards the data points in error and prints out the diagnostic:

THE NONNUMERIC (XYZ) FOUND IN RECORD N

each time such an error occurs; only after 10 such errors does STATIS give up and ask the user what to do next (see fig. 5.30c).

\*\*\*\*\*  
 \* FILE FNDEM0429 CONTAINS 112 DATA ITEMS FOR 429 RECORDS.  
 \*

\* \*TYPE A MODULE NAME AND/OR INSTRUCTIONS \*  
 \* --->? \*  
 \* STATIS \*  
 \*

\* \*TYPE LABEL FOR WHICH STATIS DESIRED -->? AGE \*  
 \*

\* STATISTICAL ANALYSIS OF AGE \*  
 \*

\* NO. OF POINTS 429 \*  
 \*

DECILE	%	CUM %	FREQ	CUM FREQ	
1	11.7	11.7	50	50	XXXXXXXXXXXXXXXXXXXXXXXXXX
2	9.3	21.0	40	90	XXXXXXXXXXXXXXXXXXXX
3	8.6	29.6	37	127	XXXXXXXXXXXXXXXXXXXX
4	12.8	42.4	55	182	XXXXXXXXXXXXXXXXXXXXXXXXXX
5	14	56.4	60	242	XXXXXXXXXXXXXXXXXXXXXXXXXX
6	14.5	70.9	62	304	XXXXXXXXXXXXXXXXXXXXXXXXXX
7	13.8	84.70	59	363	XXXXXXXXXXXXXXXXXXXXXXXXXX
8	9.1	93.80	39	402	XXXXXXXXXXXXXXXXXXXX
9	4.7	98.50	20	422	XXXXXXXXXX
10	1.6	100.10	7	429	XXX

(A)

\* MEASURES OF LOCATION \*  
 \* UNWEIGHTED MEAN 44.5618 \*  
 \* WEIGHTED MEAN 44.5618 \*  
 \* MEDIAN 46 \*  
 \*

\* MEASURES OF DISPERSION \*  
 \* STANDARD DEVIATION 12.1538 \*  
 \* STANDARD DEVIATION OF MEAN .586789 \*  
 \* RANGE 51 \*  
 \* MEAN DEVIATION 10.2193 \*  
 \* VARIANCE 147.714 \*  
 \* COEFF. OF VARIANCE 27.2740 \*  
 \*

Figure 5.30a. Results of a statistical analysis of ages produced by the STATIS module. The results are continued on the next page.

```

* * * * *
*
* TESTS FOR NON-RANDOMNESS
* NO. OF RUNS UP AND DOWN 254
* EXPECTED NO. OF RUNS 285.667
* S.D. OF NO. OF RUNS 8.71461
* MEAN SQ. SUCCESSIVE DIFF. 182.862
* MEAN SQ. SUCC DIFF/VAR 1.23795
* DIFF.S.D. OF RUNS 3.63374
*
*
* COMPUTATIONS ON DEVIATIONS
* NO. OF + SIGNS IN DEVIATIONS 226
* NO. OF - SIGNS IN DEVIATIONS 203
* NO. OF SIGN CHANGES IN DEVIATIONS + 1 159
* EXPECTED NO. OF RUNS 214.883
* S.D. OF NO. OF RUNS 10.3143
* DIFF./S.D. OF RUNS 5.41808
* TREND .013126
* S.D. OF TREND 2.275001E-4
* TREND/S.D. OF TREND 57.6976
*
*
* OTHER STATISTICS
* MINIMUM VALUE 22
* MAXIMUM VALUE 73
* BETA ONE 7.634486E-3
* BETA TWO 2.11016
* SUM OF VALUES 19117
* WEIGHTED SUM OF SQUARES 915109
* STUDENT'S T 75.9417
* NO. OF NONZERO WEIGHTS 429
* SUM OF WEIGHTS 429
*
*
* DO YOU WANT ORDERED AGE AND RANKS --->? NO
*
* * * * *

```



Figure 5.30a (concluded). The next figure shows the results that would follow if the response at [B] had been YES. See figure 5.31e for additional statistical parameter for these data as produced by the STATIS command in OMNITAB II.



The idea of allowing the statistical analysis to proceed after detecting one or more file errors is predicated on the notion that in general the omission of a few data points from a large file would not invalidate many of the results of the analysis produced by this module. The decision to allow 10 mistakes before asking the user for guidance is quite arbitrary, but since the record number in which the mistake occurs is printed, the user knows how far through the file the module has progressed and can presumably make an intelligent judgment whether to continue the analysis or stop it.

The statistical analysis produced by this module can be augmented with a graphical analysis via the STATPLOTS module. That module interfaces with OMNITAB II and has provision for generating an updated and improved statistical analysis (see fig. 5.31e).

```

* * * * *
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >statis
*   *TYPE LABEL FOR WHICH STATIS DESIRED --->? > dage
* !THE NON-NUMERIC // WAS FOUND IN RECORD 569
* !THE NON-NUMERIC // WAS FOUND IN RECORD 570
* !THE NON-NUMERIC @@ WAS FOUND IN RECORD 571
* !THE NON-NUMERIC @N WAS FOUND IN RECORD 572
* !THE NON-NUMERIC // WAS FOUND IN RECORD 573
* !THE NON-NUMERIC // WAS FOUND IN RECORD 574
* !THE NON-NUMERIC // WAS FOUND IN RECORD 575
* !THE NON-NUMERIC // WAS FOUND IN RECORD 576
* !THE NON-NUMERIC @@ WAS FOUND IN RECORD 577
* !THE NON-NUMERIC // WAS FOUND IN RECORD 578
*   *SHALL WE CONTINUE --->? >no
* * * * *

```

Figure 5.30c. This is how the STATIS module notifies the user that it ignored certain data items in carrying out a statistical analysis.

### 5.31 STATPLOTS

This module interfaces with OMNITAB II to produce four plots, developed by James J. Filliben, useful in an exploratory analysis of a column of data. There is also an option to perform a statistical analysis. The latter is an improved version of the analysis performed by the STATIS module. The following descriptions of the four plots are taken from Hogben and Peavy in reference 12.

Plot 1, in the upper left-hand corner, is a simple plot of the measurements  $X(i)$  versus the row number  $i$  in the order the measurements are entered into the worksheet. This plot may be used to detect many different patterns of nonrandomness such as trends, outliers, etc.

Plot 2, in the upper right-hand corner, is an autoregressive plot of  $X(i)$  versus  $X(i-1)$ . The plot is particularly useful for assessing lack of independence in the data. If the measurements are well behaved, the plot should not show any recognizable pattern.

Plot 3, in the lower left-hand corner, is a histogram. The histogram, designed by J. J. Filliben, is somewhat different from a conventional histogram. The interval for the horizontal axis is 0.2 standard deviations so that the horizontal axis goes from  $-5$  to  $+5$  standard deviations. Two scales are shown: one for the values of the measurements and one below it for multiples of standard deviations from the mean. Frequency is shown on the vertical axis. Information is printed below the histogram which gives the number of measurements, the value of the 0.2 standard deviation class width, the number of observations in excess of the mean  $\pm 5$  standard deviations, and the number of measurements represented by a plotted  $X$  in the histogram.

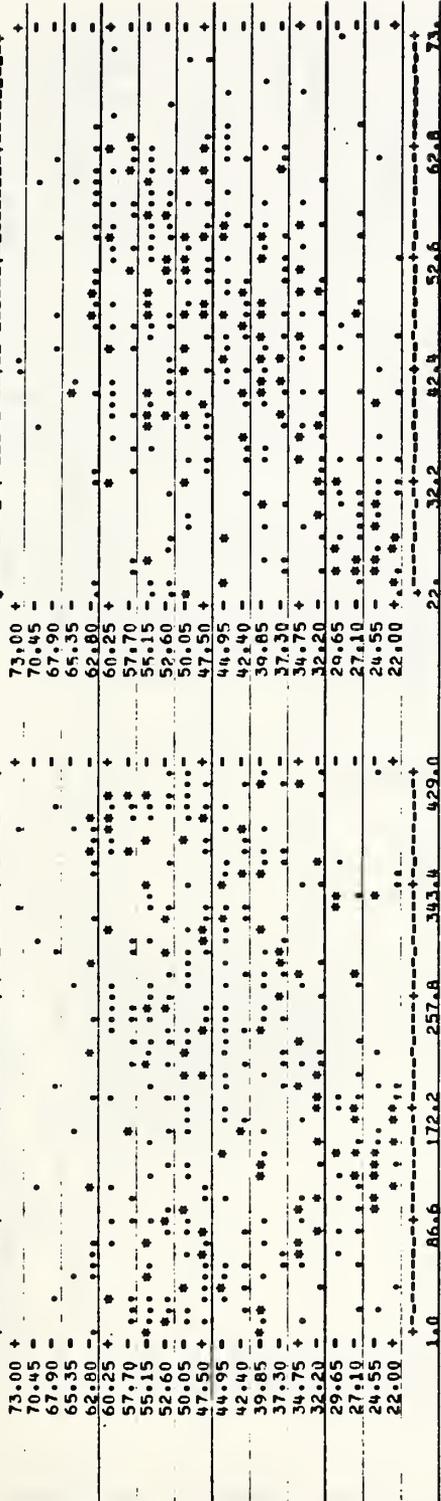
Plot 4, in the lower right-hand corner, is a normal probability plot. An assumption of a normal distribution of measurements is not satisfied if the points do not lie approximately on a straight line. Any marked curvature indicated by the points in the plot is a sign of nonnormality.

Values of the probability plot correlation coefficient, the scrawl, the mean, and the standard deviation are printed on the bottom of the plot in figures 5.31a and b. Detailed interpretation of these values and of probability plots in general is to be found in Hogben and Peavy (reference 12).

The vertical and horizontal scales for all four plots are determined by the minimum and maximum values plotted. The plotting symbol for plots 1, 2, and 4 is a period (.). The plotting symbol for plot 3 (histogram) is an  $X$ . Any  $X$  may represent 1 or more measurements depending upon the frequency indicated on the vertical axis. The maximum number of measurements represented by an  $X$  is given in the last line of information at the bottom of the histogram.

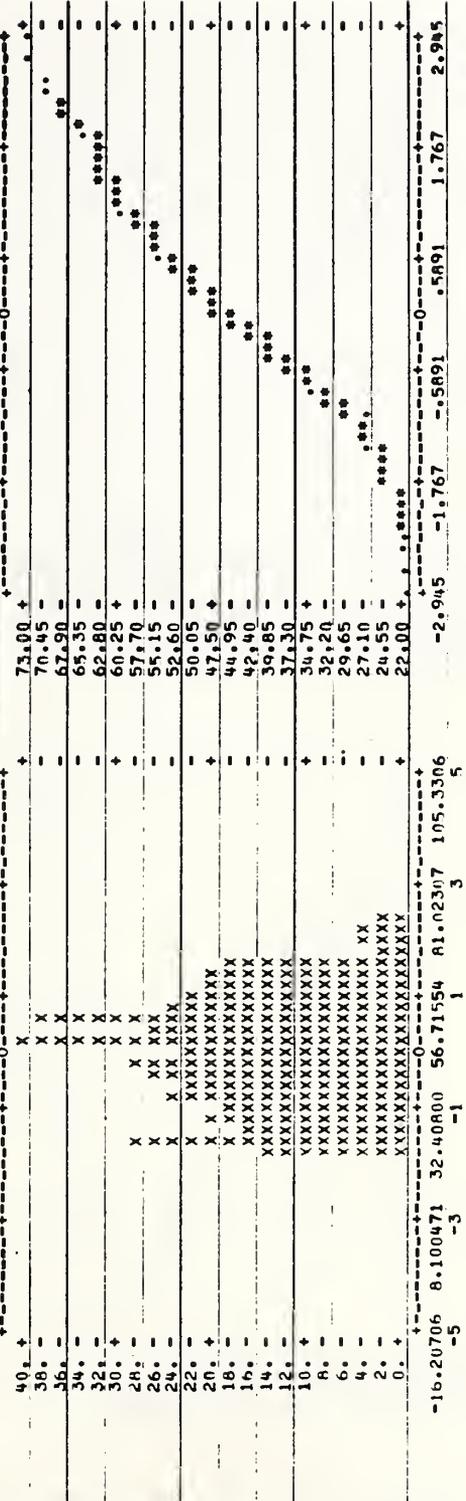
PLOT OF X(I) VERSUS I

PLOT OF X(I) VERSUS X(I-1)



HISTOGRAM

NORMAL PROBABILITY PLOT



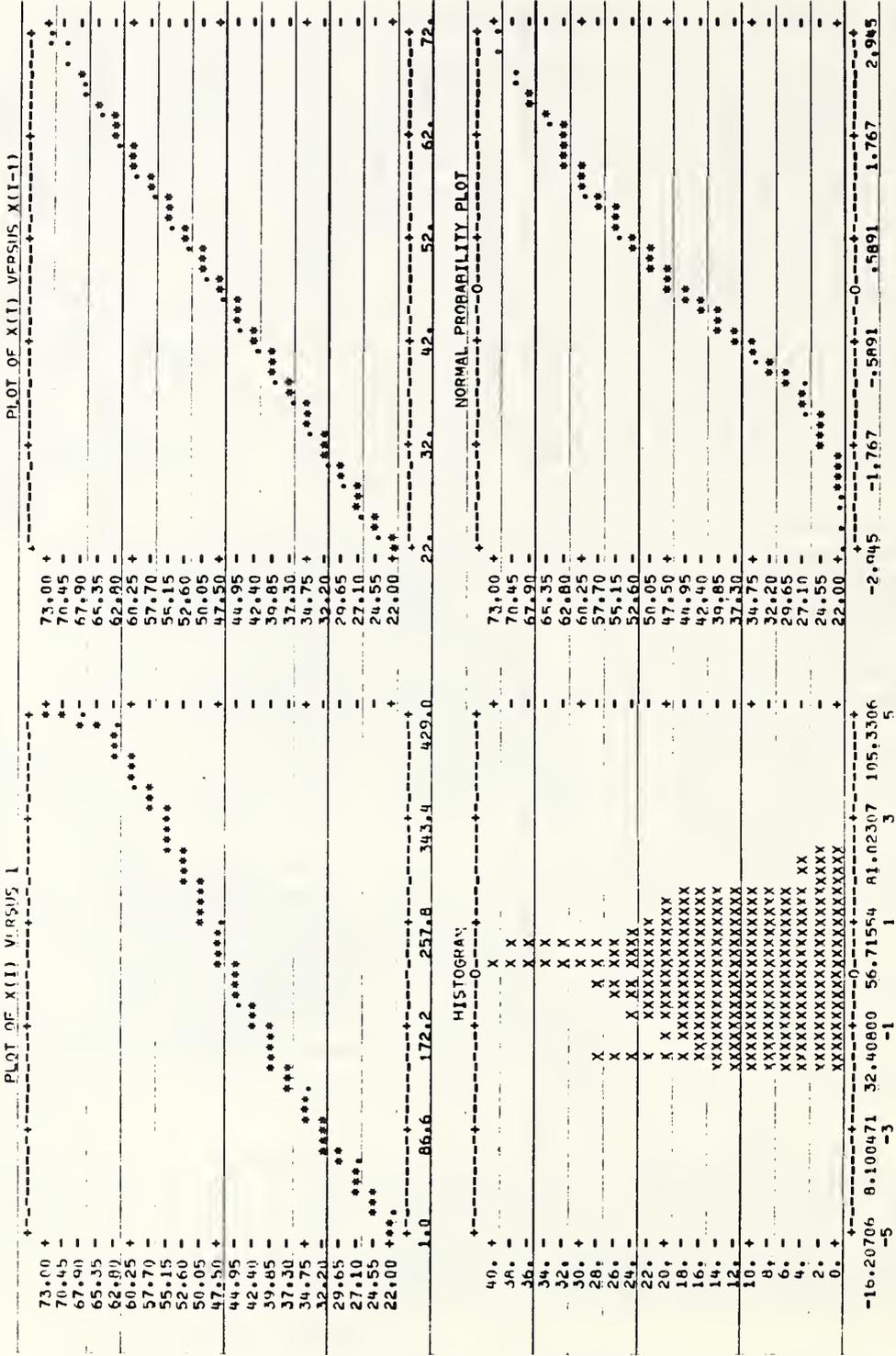


Figure 5.31b. The two plots in the upper portion of this figure demonstrate clearly that the data were not presented in random order.

NUMBER OF MEASUREMENTS = 429  
 CLASS WIDTH = .2 STANDARD DEVIATIONS = 2.4307534  
 0 OBSERVATIONS WERE IN EXCESS OF 5 STANDARD DEVIATIONS  
 ABOUT THE SAMPLE MEAN AND WERE NOT PRINTED IN HISTOGRAM  
 EACH X REPRESENTS BETWEEN 1 AND 2 MEASUREMENTS

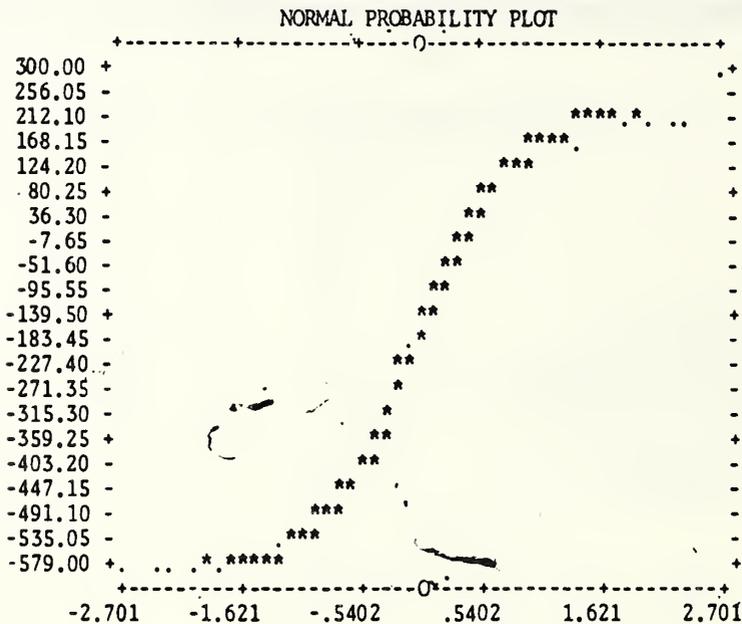
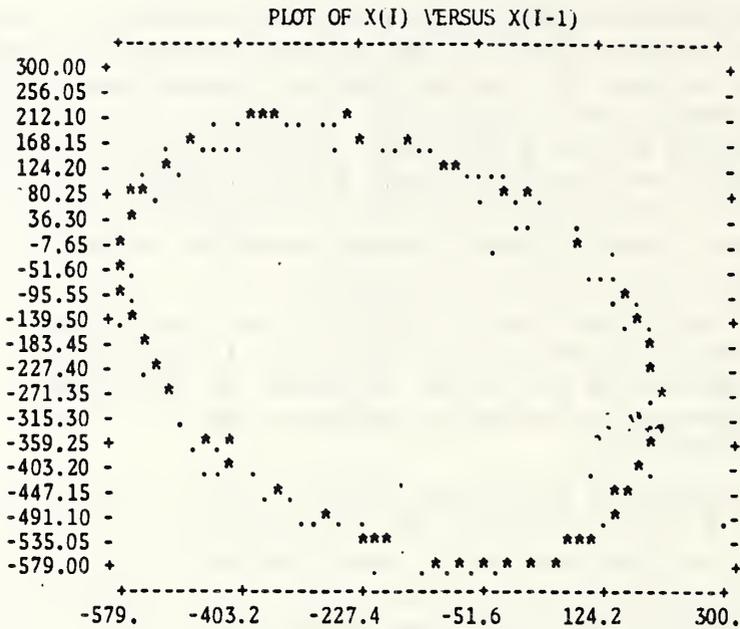
PROBABILITY PLOT CORRELATION COEFFICIENT = .9888  
 SCRAML (NE 429) = 22.000000, HE 35.000000,  
 ME 46.000000, SE 54.000000, \*E 73.000000  
 MEAN = 44.561771  
 STANDARD DEVIATION = 12.151767

It is beyond the scope of this publication to discuss the statistical foundations for these plots. We can, however, illustrate how different these plots look when the data are presented in random order and when they are arranged more systematically. To do this we have performed STATPLOTS on the ages of 429 persons arranged in fairly random order as is apparent from the plots in figure 5.31a. In figure 5.31b we show STATPLOTS of the age vector after it has been sorted. The difference in the top plots of these two figures is obvious even to the untrained eye.

In figure 5.31c, taken from reference 12, we see a still more interesting data pattern when STATPLOTS was used to analyze 200 measurements on the deflection of a steel beam. In figure 5.31d, we see how the user instructs this module to produce the STATPLOTS in the previous figures and to produce the alternate statistical analysis shown in figure 5.31e and 5.31f.

The STATPLOTS module also provides the user with the option of performing a statistical analysis via the OMNITAB II system. This produces a number of statistical parameters (see fig. 5.31e) not generated in the Omnidata STATIS module (5.30). The statistical analysis produced by the STATPLOTS module also has provision for listing, ordering, and ranking the observed data as is shown in figure 5.31f.

\* \* N O T E S \* \*



```

. PROBABILITY PLOT CORRELATION COEFFICIENT = .9540
. SCRAWL (N= 200, *= -579.00000, H= -453.00000,
. M= -162.00000, H= 94.000000, *= 300.00000)
. MEAN = -177.43500
. STANDARD DEVIATION = 277.33216

```

Figure 5.31c. This portion of a STATPLOTS output, taken from reference 12, shows the relationship of successive measurements of the deflection of a steel beam. Note how different this plot looks from the corresponding plot in the previous figures. An explanation of the statistical significance of this plot is beyond the scope of this handbook.

```

* * * * *
* *TYPE NAME FOR OMNITAB FILE --->? >stplf *
* *TYPE P(ERMANENT) OR T(EMPORARY) FOR FILE --->? >t. *
* *TYPE A TITLE FOR THIS RUN OR N(ONE) ---> *
* ? >n *
* *TYPE LABELS TO BE TRANSFERRED TO OMNITAB FILE *
* (SEQ IS A LEGITIMATE ENTRY) ---> *
* ? >yob,grade *
* YOUR WORKSHEET IS DIMENSIONED FOR 500 ROWS BY 25 COLUMNS *
* THE DATA ARE STORED IN THE WORKSHEET AS FOLLOWS *
* COLUMN LABEL *
* 1 YOB *
* 2 GRADE *
* *INPUT ANY OMNITAB COMMANDS YOU WANT PERFORMED *
* BEFORE STATPLOTS—TYPE END TO END ---> *
* ? >end *
* *TYPE LABELS OR COL NUMBERS FOR WHICH STATPLOTS DESIRED ---> *
* ? >yob,grade *
* *TYPE LABELS OR COL NUMBERS FOR WHICH STATIS IS DESIRED ---> *
* ? >yob,grade *
* *DO YOU WANT ANY MORE STATPLOTS --->? >no *
* FOLLOWING IS A LIST OF YOUR OMNITAB STATPLOTS INSTRUCTIONS: *
* 1 @ASG,A STPLFD *
* 2 @USE 7.,STPLFD *
* 3 @NBS*OMNITAB. *
* 4 OMNITAB *
* 5 DIMENSION THE WORKSHEET TO HAVE 500 ROWS AND 25 COLUMNS *
* 6 FORMAT A (2F12.0) *
* 7 READ TAPE A A INTO COLUMNS 1*** 2 *
* 8 LABEL YOB, GRADE *
* 9 STATPLOTS OF COLUMN 1 *
* 10 STATPLOTS OF COLUMN 2 *
* 11 STATISTICAL ANALYSIS OF COUNM 1 *
* 12 STATISTICAL ANALYSIS OF COLUMN 2 *
* 13 STOP *
* *DO YOU WISH TO CHANGE ANYTHING --->? >no *
* * * * *

```

Figure 5.31d. Here we have the dialogue between the user and the STATPLOTS module to produce the preceding and following figures.

```

* * * * *
*
*  OMNITAB
*
*  STATISTICAL ANALYSIS OF      CAGE                                PAGE 2
*
*  NUMBER OF MEASUREMENTS = 429, NO. OF DISTINCT MEASUREMENTS = 51
*
*  FREQUENCY DISTRIBUTION WITH 10 CLASSES OF LENGTH 5,1000000
*  50  40  37  55  60  62  59  39  20  7
*  FREQUENCY DISTRIBUTION OF LEAST SIGNIFICANT DIGIT (0,1,...,9)
*  0  0  0  0  0  0  0  0  0  0
*
*  MEASURES OF LOCATION
*  ARITHMETIC MEAN . . . . . 44.561771
*  MEDIAN . . . . . 46.000000
*  MID-RANGE . . . . . 47.500000
*  MID-MEAN (25 PERCENT TRIMMED MEAN) . . 45.120930
*
*  MEASURES OF DISPERSION
*  STANDARD DEVIATION . . . . . 12.153767
*  AS PERCENT OF MEAN (COEF. OF VAR.) . 27.273976
*  RANGE . . . . . 51.000000
*  MEAN DEVIATION . . . . . 10.219299
*  INTER-QUARTILE RANGE . . . . . 19.000000
*  VARIANCE . . . . . 147.71405
*
*  STANDARD DEVIATION OF MEAN. . . . . .58678939
*
*  TREND STATISTICS
*  SLOPE, SIGNIFICANCE LEVEL . . . . . .013126215 .005
*  QUADRATIC COEFF., SIGNIFICANCE LEVEL . 1.7407352-04 .000
*  OTHER TESTS FOR NON-RANDOMNESS
*  NUMBER OF RUNS UP AND DOWN, Z VALUE . 254 -3.634
*  MEAN SQUARE SUCCESSIVE DIFFERENCE . . 182.86215
*  MS SUCC DIFF/(VARIANCE), Z VALUE . . .61897345 -7.906
*  DEVIATIONS FROM ARITHMETIC MEAN
*  NUMBER OF - SIGNS, + SIGNS . . . . . 203 226
*  NUMBER OF RUNS, Z VALUE . . . . . 158 -5.515
*  AUTOCORRELATION COEFFICIENT . . . . . .38082922
*
*  A TWO-SIDED 95 PERCENT CONFIDENCE INTERVAL FOR THE
*  MEAN IS 43.408424 TO 45.715119
*  MEDIAN IS 44.000000 TO 47.000000 (DISTRIBUTION-FREE)
*  S.D. IS 11.396942 TO 13.034571
*
*  STATISTICAL TOLERANCE INTERVAL WITH 95 PERCENT CONFIDENCE FOR
*  50 PCT NORMAL COVERAGE IS 35.863441 TO 53.260101
*  95 PCT NORMAL COVERAGE IS 19.285748 TO 69.837794
*  99 PCT NORMAL COVERAGE IS 11.344409 TO 77.779133
*  INTERVAL FROM MIN TO MAX HAS DIST.-FREE COVERAGE 98.90
*
*  OTHER STATISTICS
*  MINIMUM . . . . . 22.000000
*  SECOND MINIMUM . . . . . 22.000000
*  MAXIMUM . . . . . 73.000000
*  SECOND MAXIMUM . . . . . 72.000000
*  (MEAN-MINIMUM)/STANDARD DEVIATION . . 1.8563604
*  (MAXIMUM-MEAN)/STANDARD DEVIATION . . 2.3398695
*  SORT(B1), SKEWNESS COEFFICIENT . . . . -0.087375473
*  B2, KURTOSIS COEFFICIENT . . . . . 2.1101557
*  LOWER QUARTILE . . . . . 35.000000
*  UPPER QUARTILE . . . . . 54.000000
*
* * * * *

```

Figure 5.31e. Here we see the result from a statistical analysis of the 429 ages performed by the STATPLOTS module via the STATIS command in OMNITAB II. It contains a number of statistical parameters that are not produced by the STATIS module discussed in section 5.30.

OBSERVATIONS

ORDERED OBSERVATIONS

I	X(I)	RANK	X(I)-MEAN	NO.	X(J)	X(J+1)-X(J)
1	40.000000	156.0	-4.561771	111	22.000000	.000000
2	62.000000	399.0	17.438229	163	22.000000	.000000
3	55.000000	332.5	10.438229	164	22.000000	.000000
4	40.000000	156.0	-4.561771	174	22.000000	.000000
5	55.000000	332.5	10.438229	183	22.000000	.000000
6	49.000000	261.0	4.438229	346	22.000000	1.000000
7	55.000000	332.5	10.438229	31	23.000000	.000000
8	52.000000	297.0	7.438229	110	23.000000	.000000
9	48.000000	248.5	3.438229	131	23.000000	.000000
10	52.000000	297.0	7.438229	145	23.000000	.000000
11	35.000000	107.5	-9.561771	146	23.000000	.000000
12	58.000000	368.0	13.438229	172	23.000000	.000000
13	39.000000	143.5	-5.561771	187	23.000000	.000000
14	56.000000	347.5	11.438229	351	23.000000	1.000000
15	43.000000	187.5	-1.561771	96	24.000000	.000000
16	41.000000	169.0	-3.561771	100	24.000000	.000000
17	52.000000	297.0	7.438229	122	24.000000	.000000
18	57.000000	358.5	12.438229	129	24.000000	.000000
19	28.000000	54.0	-16.561771	130	24.000000	.000000
20	25.000000	31.0	-19.561771	140	24.000000	.000000
21	40.000000	156.0	-4.561771	143	24.000000	.000000
22	49.000000	261.0	4.438229	191	24.000000	.000000
23	60.000000	380.0	15.438229	332	24.000000	.000000
24	55.000000	332.5	10.438229	337	24.000000	.000000
25	46.000000	218.5	1.438229	426	24.000000	1.000000
26	58.000000	368.0	13.438229	20	25.000000	.000000
27	60.000000	380.0	15.438229	98	25.000000	.000000
28	47.000000	232.5	2.438229	99	25.000000	.000000
29	67.000000	422.0	22.438229	108	25.000000	.000000
30	53.000000	311.0	8.438229	123	25.000000	.000000
31	23.000000	10.5	-21.561771	138	25.000000	.000000
32	44.000000	198.0	-5.561771	152	25.000000	.000000
33	50.000000	274.0	5.438229	212	25.000000	.000000
34	33.000000	92.5	-11.561771	331	25.000000	.000000
35	48.000000	248.5	3.438229	334	25.000000	.000000
36	49.000000	261.0	4.438229	338	25.000000	1.000000
37	50.000000	274.0	5.438229	142	26.000000	.000000
38	42.000000	178.0	-2.561771	144	26.000000	.000000
39	45.000000	209.0	1.438229	173	26.000000	.000000
40	55.000000	332.5	10.438229	180	26.000000	.000000
41	55.000000	332.5	10.438229	226	26.000000	.000000
42	48.000000	248.5	3.438229	335	26.000000	1.000000
43	46.000000	218.5	1.438229	53	27.000000	.000000
44	56.000000	347.5	11.438229	77	27.000000	.000000
45	55.000000	332.5	10.438229	109	27.000000	.000000
46	40.000000	156.0	-4.561771	119	27.000000	.000000

Figure 5.31f. Here we see the format of the optional printout of the observations in their normal order and their ranks when ordered.

## 5.32 SUMMARY

This module produces a report giving for each of the numeric data vectors in a particular file:

- a. the number of data entries (exclusive of blanks);
- b. the maximum value found;
- c. the minimum value found;
- d. the total; and,
- e. the average.

When a data file is defined into Omnidata format, space is reserved for keeping an up-to-date summary of the type discussed above and illustrated in figure 5.32a. Upon opening a file, Omnidata searches the appropriate sectors of the file and determines whether the summary exists. If it does exist, a switch is set so that when the SUMMARY module is called it need not regenerate the information, but simply print out the report as shown in figure 5.32a.

If the summary is not a part of the existing file, this module so informs the user and asks whether it should be generated. If the response to the question:

SUMMARY DOES NOT EXIST.TYPE 'YES' TO GENERATE IT--->?

is YES, the entire file is read, the summary is computed and printed as shown, and the summary is inserted into the working copy of the file.

Since the summarization is a relatively expensive operation, it can be avoided by answering 'NO' to the above question. This module has been programmed in the above manner on the assumption that a user might wish to see a summary of a file if it already exists, but may not wish to incur the cost if it does not. This is especially true when it is known that the file will be updated soon, thus rendering the summary obsolete.

When an existing file in Omnidata format is modified via SEARCH, ABRIDGE, COMPUTE, etc., during an active run, Omnidata does not update the summary until such time as the SUMMARY module is called. If SUMMARY is not called on the current file before it is saved, the SAVE module blanks out the sectors containing the summary information. Thus the summarization is deferred until such time as it is called for rather than prior to saving the file. This is done on the assumption that the file may be further modified before a summary is requested, and in any event, it costs no more to defer the summary until it is really needed.

```

*****
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? summary
*
*      *TYPE GENERATE,PRINT,OR END --->? >print
*
* LABEL      ITEMS  MAX      MIN      TOTAL      AVERAGE  VERIFY
* -----
* DIV        500    6        1        1234        2.468    *
* YOB        500    57       4        16332       32.664   *
* FC         216    99       0        6451        29.8657  ? *
* GRADE      500    17       0        4621        9.242    *
* STEP       500    10       0        2198        4.396    *
* PAY        500    36000    0        7.499638E+6  14999.3  *
* PLANT      500    650      100      145087      281.174  *
* DEPT       500    65002    10000    1.406133E+7  28122.7  *
* EODY       500    73       35       32100       64.2     *
* DEG        394    4        0        520         1.31980  *
* YR         376    73       0        13164       35.0106  ? *
* VP         500    52       12       8290        16.58    *
* INS        500    4        1        931         1.862    *
* HC         232    55       0        349         1.50431  ? *
* RET        500    4        1        563         1.126    *
* NOAC       500    91800    10111    4.093189E+7  81863.8  *
* TLWOP      500    40       0        184         .368     *
* CLWOP      500    3810     0        19567       39.134   *
* CAWOL      500    112      0        259         .518     *
* CSL        500    500      0        15276       30.552   *
* NP         428    0        0        0           0        ? *
* NHSA       352    7772     0        152644      433.648  ? *
* LWOPSA     297    0        0        0           0        ? *
* MR         496    9        0        849         1.71169  ? *
* SK         459    9        0        682         1.48584  ? *
* HAWAE     482    120      0        1077        2.23444  ? *
* SCDY       500    73       0        29640       59.28    *
* C          500    63       0        1936        3.872    *
* AGE        500    73       20       22168       44.336   *
*
*      *TYPE GENERATE,PRINT,OR END --->? >stop
* PROGRAM STOPPED.
*****

```

Figure 5.32a. Here we see a summarization of all of the numeric data vectors in the file FNDEMO. The question mark in the verify column alerts us that the numbers in that row are based on a count different than 500. The lesser count results either from missing data or from alphabetic characters which may or may not be legitimate.

### 5.33 SURVEY

In many files derived from answers to questionnaires or other types of surveys, the data are represented by one- or two-digit numbers or by the letters A-Z. The proper analysis of such data requires, among other things, facilities for bivariate and univariate analysis. Facilities for bivariate analysis are provided in the CROSSTAB module.

This module permits a complete univariate analysis of all of the data vectors in a single pass through the file when the data entries consist of one- or two-digit numerics or single alphabetic characters. Figure 5.33a shows typical data records of files amenable to analysis by the SURVEY mode. In figure 5.33b we show the primary report produced by the SURVEY module when applied to a file containing information on motor vehicle accidents. The numbers in the body of the table in figure 5.33b show for each data vector (represented by a row and identified by its label) how many records contained the specific value (entry) indicated at the top of the column. Thus 57 records contained a 1 in the data vector labeled TSEQ. Similarly, 42 records contained a 2, 1 record contained a 3, etc. The last page of the automatic output from this module is shown in figure 5.33c. There we see that the table extends to 99 in 3 of the data fields. Had all of the data fields been defined as single numeric characters, the main table would contain 10 columns for the digits 0-9, 1 column for blanks, and 1 column for errors. In this instance an error is recorded when an alphabetic or graphic character appears in a normally numeric position.

If the SURVEY is performed on alphabetic entries in a file such as is shown in figure 5.33a, it is valid only for single character data fields. In that case the main table consists of 26 columns (A-Z), plus 1 column to record blanks and 1 column for errors. In this instance the presence of any numeric or graphic character is considered an error.

While the restriction of one character for alphabetic data and two for numerics may seem unduly limiting, this is, in fact, a very powerful analysis tool for many diverse files as the analysis is performed on dozens of data vectors in a single pass through the file. Furthermore, the user is reminded that data vectors can be analyzed via the TALLY module discussed later. While that module has no limitation on character length, it is possible to perform the TALLY operation on only three data vectors on a single pass through the file. It is beyond the scope of this publication to explain the reasons for the restraints in the TALLY module as opposed to these in the SURVEY module.

In the course of preparing the analysis shown in figure 5.33b, the SURVEY module keeps totals and computes averages which are printed (see fig. 5.33c) at the end of the main survey table. This module also produces a version of the main table in the form of percentages if desired (see fig. 5.33d). This format is at least as useful and is often a more desirable presentation of the data. In figure 5.33e we see how this module reports the incidence of alphabetic characters in the file.



ANALYSIS OF DATA FOUND IN FILE DELVIOO (ACTUAL VALUES)											
ENTRY =	0	1	2	3	4	5	6	7	8	9	10
1 YEAR	0	0	0	0	0	0	0	0	0	0	0
2 MONTH	0	100									
5 TSEQ	0	57	42	1							
6 INV-AG	0	49	20	0	1	0	6	0	24		
7 COFA	0	85	6	9							
8 CLASS	0	1	4	2	66	13	0	1	0	12	1
9 SEVER	0	0	19	81							
10 TROOP	0	20	0	6	3	4	35	2	2	4	24
11 TOWN	44	0	0	0	0	0	0	0	0	0	0
12 DSEX	0	61	28	7	4						
13 DAGE	4	0	0	0	0	0	0	0	0	0	0
14 DOCCUP	4	15	2	12	0	5	1	35	5	2	9
15 COD	16	67	5	9	3						
16 PROX	4	79	2	12	0	3					
18 LTYPE	4	71	7	0	1	10	0	3	4		
19 DEXP	4	0	0	2	66	4	4				
20 DEDEC	4	48	40	1	3	4					
22 DPHYS	4	0	0	0	1	0	91	4			
23 DGLASS	4	24	72								
24 DRINK	4	87	1	1	3	1	3				
25 DTEST	4	0	0	0	0	0	1	96			
26 BAC	99	0	0	0	0	0	0	0	0	0	0
27 HANDR	0	98	0	2							
28 PIVOL	0	1	8	0	3	4	1	0	0	2	0

Figure 5.33b. A portion of the first page of output from the SURVEY module when applied to 100 records of a file containing information on automobile accidents.

```

*****
*
*                               ANALYSIS OF DATA FOUND IN FILE
*  ENTRY =                      16    17    18    19    20    21    22    23
* -----
*  1 YEAR                        0     0     0     0     0     0     0     0
*  2 MONTH
*  5 TSEQ
*  6 INV-AG
*  7 COFA
*  8 CLASS
*  9 SEVER
* 10 TROOP
* 11 TOWN                        0     0     0     0     0     0     0     0
* 12 DSEX
* 13 DAGE                        0     5     4     3     5     5     5     4
* 14 DOCCUP
* 15 COD
* 16 PROX
* 18 LTYPE
* 19 DEXP
* 20 DEDEC
* 22 DPHYS
* 23 DGLASS
* 24 DRINK
* 25 DTEST
* 26 BAC                        0     0     0     0     0     1
* 27 HANDR
* 28 PIVOL                      2     0     0    11     3     1     0     9
* 30 NOPIV
* 31 VYEAR                      0     0     0     0     0     0     0     0
* 32 MAKE                        2     1     0     2     0     0     2     0
* 33 BODY                        0     0     0     0     0     0     0     0
* 34 VSPEC
*
* 40 DAMLOC
* 41 SPEED
*
*                               ANALYSIS OF DATA FOUND IN FILE
*  ENTRY =                      32    33    34    35    36    37    38    39
* -----
*  1 YEAR                        0     0     0     0     0     0     0     0
*  2 MONTH
*****

```

Figure 5.33b (continued). Here we see portions of the second and third pages of the summary. The number of columns displayed per page depends on the width setting (100 in this case). Note that data items 3, 4, 17, 21, 29, 42, and 43 are missing. Item 4 is alphabetic and is recorded in figure 5.33e. The other data fields are longer than two characters and are therefore not tallied.

```

*****
* ANALYSIS OF DATA FOUND IN FILE DELVIOO (ACTUAL VALUES) *
* ENTRY =          96          97          98          99 *
* -----*-----*-----*-----*-----*
* 1 YEAR
* 2 MONTH
* 5 TSEQ
* 6 INV-AG
* 7 COFA
* 8 CLASS
* 9 SEVER
* 10 TROOP
* 11 TOWN
* 12 DSEX
* 13 DAGE          0          0          0          7
* DDCCUP
* 15 COD
* 16 PROX
* 18 LTYPE
* 19 DEXP
* 20 DEDEC
* 22 DPHYS
* 23 DGLASS
* 24 DRINK
* 25 DTEST
* 25 BAC
* 27 HANDR
* 28 PVIOL        0          0          0          51
* 30 NOPIV
* 31 VYEAR        0          0          0          3
* 32 MAKE
* 33 BODY
* 34 VSPEC
* 35 VVOBS
* 36 VDEF
* 37 DACTION
* 38 MACTION
* 39 DAMAGE
* 40 DAMLOC
* 41 SPEED
*****

```

Figure 5.33b (concluded). Here we see the final page of the primary survey table. Note that zeros are suppressed in each data item after the highest entry has been tabulated. An entry of 99 probably denotes missing information.

As illustrated thus far, the output of the SURVEY module is a great boon to the file manager as it provides him with a comprehensive and more or less readily comprehensible picture of the anatomy of his data base. Aside from the information contained in the body of the main tables it provides totals and averages where they are meaningful. In the data used in the figures 5.33b. et seq. the averages shown have meaning in the data vectors showing age (DAGE), continuous driving time immediately prior to the accident (DTIME), the age of the vehicle (VYEAR), etc. Totals would be meaningful if the file contained data on repair costs, etc.

The results shown in figure 5.33b are immediately comprehensible only if one knows what the entries 1, 2, 3, or 4 mean in the TSEQ vector or what the values of 1, 2, or 3, etc., mean when applied to the classification (CLASS) of an accident. Lacking that knowledge, the tabulated results leave more to be desired. The SURVEY module is, indeed, able to supply the analysis in more explicate form if the file builder has taken the trouble to build into the file a data dictionary giving the meaning of the digits found in each of the data vectors. A discussion of the structure of the data dictionary is to be found in the discussion of the utility module DICTIONARY (sec. 6.4).

After completing the tables shown in the figures thus far, the SURVEY module checks to see if a dictionary is contained in the main file. If a dictionary is found, an affirmative response to the question:

\*DO YOU WISH TO HAVE A DECODED REPORT--->?

produces a report in the form shown in figure 5.33f. In this format the report is immediately intelligible without recourse to other documents.

At this stage, or earlier if there is no dictionary file, the user is given the opportunity to write the data on a file to be used by OMNITAB II or other processors.

We return now to the initiation of a SURVEY run to explain the options provided for when the user is asked to:

\*TYPE LABELS FOR WHICH SURVEY IS DESIRED--->?

At this point the user can type:

- a) a list of labels (TSEQ, INV-AG, COFA, CLASS, etc.);
- b) a list of numbers synonymous with the labels, 1, 2, 3, 4, 12, 37, etc.;
- c) a number pair indicating a range of labels (5-24);
- d) combinations of the above; or
- e) ALL

Regardless of the above response, the system separates the numeric from

the alphabetic data vectors and produces two sets of tables if required. If the above instructions contain data vectors which are defined as extending over more than two numeric positions (or more than one alphabetic) they are ignored in the survey.

Since the computation involved in generating a survey report is extensive, provision has been made to store the results on a file when desired. Storage is achieved if the word STORE appears in the SURVEY request as follows.

\*TYPE A MODULE NAME AND/OR INSTRUCTIONS  
 --->? >width,100,store,survey

When this is the case, the user is immediately told the name of the output file upon entering SURVEY.

```

*****
*           LABEL             MEAN             TOTAL           *
* -----
*      1 YEAR                75             7500           *
*      2 MONTH                1             100            *
*      5 TSEQ                 1.44           144            *
*      6 INV-AG               3.21           321            *
*      7 COFA                 1.24           124            *
*      8 CLASS                4.69           469            *
*      9 SEVER                2.81           281            *
*     10 TROOP                5.86           586            *
*     11 TOWN                 30.14          3014           *
*     12 DSEX                 1.54           154            *
*     13 DAGE                 35.94          3594           *
*-----
*     31 VYEAR               70.77          7077           *
*     32 MAKE                 18.69          1869           *
*     33 BODY                 4.77           477            *
*     34 VSPEC                 0              0              *
*     35 VVOBS                 0              0              *
*     36 VDEF                  1             100            *
*     37 DACTION              2.98           298            *
*     38 MACTION              .5             50             *
*     39 DAMAGE               1.87           187            *
*     40 DAMLOC               8.3            830            *
*     41 SPEED                4.56           456            *
*****
  
```

Figure 5.33c. The totals and averages produced by this module have less validity in this file than in those where more of the data items represent quantitative rather than qualitative information.

DO YOU WANT THE TABLES GIVEN IN PERCENTAGES -->? >YES

ANALYSIS OF DATA FOUND IN FILE DELVIOO (IN PERCENT)

ENTRY =	0	1	2	3	4	5	6	7	8	9	10
1 YEAR	0	0	0	0	0	0	0	0	0	0	0
2 MONTH	0	100									
5 TSEQ	0	57.00	42.0	1.0							
6 INV-AG	0	49.0	20.0	0	1.0	0	6.0	0	24.0		
7 COFA	0	85.00	6.0	9.0							
8 CLASS	0	1.0	4.0	2.0	66.00	13.0	0	1.0	0	12.0	1.0
9 SEVER	0	0	19.0	81.00							
10 TROOP	0	20.0	0	6.0	3.0	4.0	35.0	2.0	2.0	4.0	24.0
11 TOWN	44.0	0	0	0	0	0	0	0	0	0	0
12 DSEX	0	61.0	28.0	7.0	4.0						
13 DAGE	4.0	0	0	0	0	0	0	0	0	0	0
14 DOCCUP	4.0	15.0	2.0	12.0	0	5.0	1.0	35.0	5.0	2.0	9.0
15 COD	16.0	67.00	5.0	9.0	3.0						
16 PROX	4.0	79.00	2.0	12.0	0	3.0					
18 LTYPE	4.0	71.00	7.0	0	1.0	10.0	0	3.0	4.0		
19 DEXP	4.0	0	0	2.0	86.00	4.0	4.0				
20 DEDEC	4.0	48.0	40.0	1.0	3.0	4.0					
22 DPHYS	4.0	0	0	0	1.0	0	91.00	4.0			
23 DGLASS	4.0	24.0	72.00								
24 DRINK	4.0	87.00	1.0	1.0	3.0	1.0	3.0				
25 DTEST	4.0	0	0	0	0	0	1.0	95.00			
26 BAC	99.00	0	0	0	0	0	0	0	0	0	0

Figure 5.33d. Here we see a table that shows, for each data item, the percentages of the records which contained the number indicated at the top of the column. These numbers came out as integers because the file contained exactly 100 records.



```

*****
*                                     *
*                               INVESTIGATING AGENCY                               *
*                                     *
* STATE POLICE 49 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
* COUNTY POLICE 20 XXXXXXXXXXXXX *
* BRIDGE POLICE 0 *
* TOWN POLICE (LESS THAN 2500) 1 *
* TOWN POLICE (2501-10000) 0 *
* TOWN POLICE (10001-25000) 6 XXX *
* TOWN POLICE (25001-50000) 0 *
* TOWN POLICE (OVER 50000) 24 XXXXXXXXXXXXXXX *
*                                     *
*                               COUNTY                                               *
*                                     *
* NEWCASTLE 85 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
* MERCER 6 XXX *
* SUSSEX 9 XXXX *
*                                     *
*                               ACCIDENT CLASSIFICATION                               *
*                                     *
* OVERTURN NON-COLLISION 1 *
* M.V. ON OTHER ROADWAY 4 XX *
* PEDISTRIAN 2 X *
* M.V. IN TRANSPORT 66 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
* PARKED VEHICLE 13 XXXXXX *
* RR TRAIN 0 *
* PEDACYCLIST 1 *
* ANIMAL 0 *
* FIXED OBJECT 12 XXXXXX *
* OTHER OBJECT 1 *
*****

```

Figure 5.33f. A portion of an optional output from the SURVEY module which shows explicitly the meanings of the column heading in figure 5.33b. The numbers in the central column are those which appear in rows 6, 7, and 8 of the figure 5.33b. The order in which the information is displayed is determined by the file builder when he prepares the dictionary file (see sec. 6.4) from which the above text is obtained.

## 5.34 TALLY

This module produces a frequency distribution and associated histograms for data entries in up to three data vectors at a time (in a single pass through the file). Actual frequency of occurrence and percentage of occurrence are always tabulated, while cumulative frequencies and cumulative percentages are tabulated when the format allows (when the vectors contain numeric data or when they contain less than 16 alphabetic characters).

The operation of TALLY upon numeric data vectors, shown in figure 5.34a, is achieved simply by naming the data vectors we wish to have tallied. Here we see a complete printout including percentage and cumulative percentage distribution of the entries in each of the designated data items. The cumulative percentages are meaningful in the case of YOB (year of birth) and are of doubtful value in the cases of PROFS and TOA unless the order of the numbers has some special significance. In any case, it is the underlying philosophy of Omnidata not to burden the user with questions and decisions in order to save a trivial amount of computation. Where these numbers are useful, they are available; where they have no important meaning, they can be ignored.

When we wish to perform a tally on an alphabetic data vector, TALLY provides a number of interesting alternatives. In figure 5.34b, we see a number of variants in which the module is instructed to perform the tally upon the first 20 characters of the data in the TITLE field, upon the first word of the field, and then on the first character of the TITLE field. In a suitably structured data file this simple feature allows for a summarization at a variety of levels difficult to achieve otherwise. Note that when the number of characters is small and there is room for it, the module prints out the cumulative frequency and cumulative percentage.

The truncation in terms of characters or words proceeded from the left in the examples shown. TALLY can also perform these truncations from the right. This feature is useful when a field contains both the city and state as it makes it possible to perform a TALLY operation on states. When applied to a vector of words this module can tally the occurrences of word endings. If this module were asked to tally an Italian word list one character from the right, we would have a statistical analysis of the terminal letters.

In figure 5.34e we see how this module formats the output when the data being tallied is longer than the 29 characters allotted by this module.

The feature of specifying the number of characters to be used in building up the frequency was incorporated originally for handling alphabetic data items. It can also be applied to numeric data with interesting and even important consequences. For example, if we perform a tally upon the PAY vector and specify a single character, we get an automatic coding of salaries in steps of \$10,000. If we use two characters,

we get a finer breakdown into \$1,000 intervals, etc. An example of such a run is shown in figure 5.34c.

Another variant in how the TALLY module is instructed concerns where in the data field to begin the tally. If one is tallying an entire field or a specific number of characters in that field, START AT X instructs the module to begin the tally with the character in position X. If the user has indicated a specific number of words to be tallied, on the other hand, the inclusion of START AT X refers to which word to start with.

The TALLY module also asks the user to

\*TYPE ANY SEPARATORS YOU WISH RECOGNIZED, OR N(ONE)-->? >

This feature is useful, for example, in a file in which one of the data vectors is keywords. If the keywords in the field are separated by semicolons and that is entered in response to the above question, then each keyword will be tallied individually.

\* \* \* N O T E S \* \* \*

```

*****
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   -->? >tally
*
*   (A) *TYPE LABELS (UP TO 3) TO BE TALLIED: FOLLOWED BY 'END'
*   -->? >toa,dept,eody
*   *TYPE ANY SEPARATORS YOU WISH RECOGNIZED. OR N(ONE)-->
*   ? > none
*
*   *DO YOU WANT THE TALLY OF TOA SORTED
*   (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*   OR (DESCEND)ING ORDER -->? >alpha
*   *SHALL WE NORMALIZE -->? >yes
*   *HOW MANY OF THE 8 ENTRIES DO YOU WANT PRINTED--->? >all
*
*           TALLY OF TOA
*
*   TOA      CUM %      % CUM  FREQ  FREQ
*   -----
*   1         74.60      74.60   320   320   XXXXXXXXXXXXXXXXXXXX
*   2         87.2       12.6    374    54    XXX
*   4         87.9       .7      377    3
*   5         92.80      4.9    398    21    X
*   6         94.2       1.4    404    6
*   7         94.60      .5     406    2
*   8         96.80      1.2    411    5
*   9         100        4.2    429    18    X
*
*   ? >
*
*   *DO YOU WANT THE TALLY OF DEPT SORTED
*   (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*   OR (DESCEND)ING ORDER -->? >alpha
*   *SHALL WE NORMALIZE -->? >yes
*   *HOW MANY OF THE 185 ENTRIES DO YOU WANT PRINTED--->? >1
*
*           TALLY OF DEPT
*****

```

Figure 5.34a. This module can perform a tally of information found in as many as three data vectors in one pass through the file, as we see at [A]. The response at [B] becomes necessary if the field being tallied contains more than one item and if each item is separated by a given symbol. For example, a field of Keywords may have multiple entries separable on semicolons. After each tally, as at [C], the program pauses to allow for positioning of the paper after which a response (any character or carriage return) will be required to signal the program to produce the next tally. At [D] we see the utility of informing the user how many lines there are in tally. In this case it turns out that DEPT was the wrong item so we list only one line.

```

*DO YOU WANT THE TALLY OF EODY SORTED
(ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
OR (DESCEND)ING ORDER --->? >alpha
*SHALL WE NORMALIZE --->? >yes
*HOW MANY OF THE 33 ENTRIES DO YOU WANT PRINTED--->? >all
TALLY OF EODY

```

EODY	CUM %	% CUM	FREQ	FREQ	
35	.2	.2	1	1	
36	.5	.2	2	1	
40	1.2	.7	5	3	X
41	1.9	.7	8	3	X
42	2.3	.5	10	2	X
43	3.3	.9	14	4	XX
44	3.7	.5	16	2	X
45	4.2	.5	18	2	X
46	5.1	.9	22	4	XX
47	5.8	.7	26	3	X
48	7.2	1.4	31	6	XX
49	8.2	.9	35	4	XX
50	8.6	.5	37	2	X
51	11.9	3.3	51	14	XXXXXX
52	14	2.1	60	9	XXXX
53	14.2	.2	61	1	
54	15.9	1.6	68	7	XXX
55	17.5	1.3	76	7	XXX
56	19.1	1.6	82	7	XXX
57	21.2	2.1	91	9	XXXX
58	22.8	1.6	98	7	XXX
59	24	1.2	103	5	XX
60	27.7	3.7	119	16	XXXXXX
61	31.2	3.6	134	15	XXXXXX
62	33.1	1.9	142	6	XXX
63	37.5	4.4	161	19	XXXXXXXXXX
64	41.3	3.7	177	16	XXXXXX
65	46.2	4.9	198	21	XXXXXXXXXX
66	51.3	5.1	220	22	XXXXXXXXXX
67	56.2	4.9	241	21	XXXXXXXXXX
68	61.8	5.6	265	24	XXXXXXXXXX
69	71.30	9.6	306	41	XXXXXXXXXXXXXXXXXXXX
70	77.4	6.1	332	26	XXXXXXXXXXXXXXXX
71	84.10	6.8	361	29	XXXXXXXXXXXXXXXX
72	93.9	9.8	403	42	XXXXXXXXXXXXXXXXXXXX
73	100	6.1	429	26	XXXXXXXXXXXXXXXX

Figure 5.34a (concluded).

```

*****
*      *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*      -->? >tally
*
*
*      *TYPE LABLES (UP TO 3) TO BE TALLIED: FOLLOWED BY 'END'
*      -->? >title,20 char
*      MORE OR END -->? >title,1 word
*      *MORE OR END -->? >title,1 char
*      *TYPE ANY SEPARATORS YOU WISH RECOGNIZED, OR N(ONE) -->?
*      ? >none
*
*      *DO YOU WANT THE TALLY OF TITLE SORTED
*      (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*      OR (DESCEND)ING ORDER -->? >alpha
*      *SHALL WE NORMALIZE -->? >no
*      *HOW MANY OF THE 62 ENTRIES DO YOU WANT PRINTED--->? >10
*      TALLY OF TITLE
*
*      ITEM                                %      FREQ
*      -----
*      MATHEMATICIAN                      1      1      X
*      RESEARCH CHEMIST                    2      2      XX
*      ADMINISTRATIVE AID                  2      2      XX
*      ADMINISTRATIVE ASSIS                2      2      XX
*      ADMINISTRATIVE OFFIC                1      1      X
*      ARCHITECT                           1      1      X
*      AREA PROGRAM                        1      1      X
*      BUILDING TECHNOLOGIS                1      1      X
*      CHEMICAL ENGINEERING                1      1      X
*      CHEMIST                              2      2      XX
*      *MORE -->? >yes
*      CHIEF RESEARCH                      1      1      X
*      CHIEF, PHOTOCHEMISTR
*      CHIEF, PHYSICAL                     1      1      X
*      CHIEF, SURFACE                      1      1      X
*      CLERK STENOGRAPHER                  1      1      X
*      COMPUTER PROGRAMMER                 1      1      X
*****

```

Figure 5.34b. In this application, we ask at [A] for three tallies on the same data vector to illustrate the facility to tally on one or more words or one or more characters. Had we instructed the module at [B] to recognize commas, the presence of the comma in certain lines at [C] would not have produced the situation at [A] in the next figure.

```

*****
*
* *DO YOU WANT THE TALLY OF TITLE SORTED
* (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
* OR (DESCEND)ING ORDER --->? >alpha
*
* *SHALL WE NORMALIZE --->? >no
*
* *HOW MANY OF THE 40 ENTRIES DO YOU WANT PRINTED--->? >20
*
* TALLY OF TITLE
*
* ITEM % FREQ
* -----
* 3.1 3 XXX
* ADMINISTRATIVE 5.1 5 XXXXX
* ARCHITECT 1 1 X
* AREA 1 1 X
* BUILDING 1 1 X
* CHEMICAL 1 1 X
* CHEMIST 2 2 XX
* CHIEF 1 1 X
* CHIEF, 3.1 3 X
* CLERK 1 1 X
* COMPUTER 3.1 3 XXX
* CONFERENCE 1 1 X
* CONSULTANT 1 1 X
*
*
* *SHALL WE NORMALIZE --->? >no
*
* *HOW MANY OF THE 15 ENTRIES DO YOU WANT PRINTED --->? >all
*
* TALLY OF TITLE
*
* TITLE CUM % % CUM FREQ FREQ
* -----
* 3.1 3.1 3 3 XXX
* A 10.2 7.1 10 7 XXXXXXXX
* B 11.2 1 11 1 X
* C 24.8 13.3 24 13 XXXXXXXXXXXXXXXX
* D 25.3 1 25 1 X
* E 35.7 10.2 35 10 XXXXXXXXXXXXX
* F 38.8 3.1 38 3 XXX
* G 40.8 2 40 2 XX
* I 41.8 1 41 1 X
* M 49 7.1 48 7 XXXXXXXX
* O 51 2 50 2 XX
* P 80.60 29.6 79 29 ++++++
* R 89.80 9.2 88 9 XXXXXXXXX
* S 99 9.2 97 9 XXXXXXXXX
* T 100 1 98 1 X
*
*****

```

Figure 5.34b (concluded). The listings at [A] would have been combined had we instructed the module to recognize (really ignore) the comma.

```

*****
* *TYPE LABELS (UP TO 3) TO BE TALLIED: FOLLOWED BY 'END' *
* -->? >pay,1 char *
* *MORE OR END -->? >pay,2 char *
* *MORE OR END -->? >pay,3 char *
* *TYPE ANY SEPARATORS YOU WISH RECOGNIZED, OR N(ONE)--> *
* ? >n *
* *DO YOU WANT THE TALLY OF PAY SORTED *
* (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING *
* OR (DESCEND)ING ORDER -->? >alpha *
* *SHALL WE NORMALIZE -->? >yes *
* *HOW MANY OF THE 4 ENTRIES DO YOU WANT PRINTED--->? >all *
* TALLY OF PAY *
*
* PAY CUM % % CUM FREQ FREQ
* -----
* 0 23.1 23.1 99 99 XXXXXXXXXXXX
* 1 61.8 38.7 265 166 XXXXXXXXXXXXXXXXXXXX
* 2 89 27.3 382 117 XXXXXXXXXXXXXX
* 3 100 11 429 47 XXXXXX
*
* ? >
*
* *DO YOU WANT THE TALLY OF PAY SORTED *
* (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING *
* OR (DESCEND)ING ORDER -->? >alpha *
* *SHALL WE NORMALIZE -->? >yes *
* *HOW MAY OF THE 33 ENTRIES DO YOU WANT PRINTED--->? >all *
* TALLY OF PAY *
*
* PAY CUM % % CUM FREQ FREQ
* -----
* 04 .7 .7 3 3 XX
* 05 3.5 2.8 15 12 XXXXXXXX
* 06 10.7 7.2 46 31 XXXXXXXXXXXXXXXXXXXX
* 07 13.3 2.3 57 11 XXXXXX
* 08 19.1 5.8 82 25 XXXXXXXXXXXXXXXXXXXX
* 09 23.1 4 99 17 XXXXXXXXXXXX
* 10 28.4 5.4 122 23 XXXXXXXXXXXXXXXX
* 11 33.1 4.7 142 20 XXXXXXXXXXXXX
* 12 35 1.9 150 8 XXXX
* 13 41.7 6.8 179 29 XXXXXXXXXXXXXXXXXXXX
* 14 45.0 4.2 197 18 XXXXXXXXXXXX
*****

```

Figure 5.34c. Here we see at [A] how tallying on first one character, then two characters, etc., provides a useful summarization of salary distribution. At [B] we have the distribution in steps of \$10,000, at [C] in steps of \$1,000.

```

* * * * *
* 29      89      .9      382      4      XX
* 30     91.60    2.3     393     11     XXXXXX
* 31      93      1.4     399      6      XXX
* 32     94.2     1.2     404      5      XXX
* 33     95.60    1.4     410      6      XXX
* 34     96.7     1.2     415      5      XXX
* 35     97.7     .9      419      4      XX
* 36     100     2.3     429     10     XXXXX

```

? >

\*DO YOU WANT THE TALLY OF PAY SORTED  
(ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING  
OR (DESCEND)ING ORDER --->? >descend

(D)

\*SHALL WE NORMALIZE --->? >yes

\*HOW MANY OF THE 95 ENTRIE DO YOU WANT PRINTED --->? >15  
TALLY OF PAY

```

* PAY      CUM %      % CUM  FREQ  FREQ
* -----
* 216     5.6        5.6    24    24    XXXXXXXXXXXXXXXXXXXXX
* 189     8.9        3.3    38    14    XXXXXXXXXXXXX
* 229    11.9        3      51    13    XXXXXXXXXXXXX
* 068    14.9        3      34    13    XXXXXXXXXXXXX
* 268    17.9        3      77    13    XXXXXXXXXXXXX
* 304    20.5       2.6    88    11    XXXXXXXXXXXXX
* 168    23.1       2.6    99    11    XXXXXXXXXXXXX
* 261    25.6       2.3   110    11    XXXXXXXXXXXXX
* 061    28.2       2.3   121    11    XXXXXXXXXXXXX
* 223    30.5       2.3   131    10    XXXXXXXXX
* 360    32.9       2.3   141    10    XXXXXXXXX
* 054    35.2       2.3   151    10    XXXXXXXXX
* 172    37.3       2.1   160     9    XXXXXXXX
* 153    39.4       2.1   169     9    XXXXXXXX
* 044    41.9       1.9   177     9    XXXXXXXX

```

\*MORE --->? >no

(E)

Figure 5.34c (concluded). Here we tell this module at [D] to arrange this tally not on the salary but in decreasing order by frequency. Thus we see clearly that most of the people in this file earned \$21,600 per year. In this instance the cummulative frequencies and cummulative percentages tell a different story than those above. Thus we see at [E] that slightly over 25% of the staff salaries fall into only 8 of the 95 salary levels.

```

* * * * *
*   *DO YOU WANT THE TALLY OF DOCCUP SORTED
*   (ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*   OR (DESCEND)ING ORDER -->? >descend
*   *SHALL WE NORMALIZE -->? >yes
*   *HOW MANY OF THE 13 ENTRIES DO YOU WANT PRINTED--->? >all
*   TALLOF OF DOCCUP
*

```

DOCCUP	CUM %	% CUM	FREQ	FREQ	
10	35.3	35.3	200	200	XXXXXXXXXXXXXXXXXXXX
07	55.7	20.5	316	116	XXXXXXXXXXXX
00	74.30	18.5	421	105	XXXXXXXXXXXX
03	82.7	8.5	469	48	XXXX
08	88.5	5.8	502	33	XXX
11	92.4	3.9	524	22	XX
06	94.7	2.3	537	13	X
05	96.30	1/3	546	9	X
12	97.4	1.1	552	6	X
09	98.2	.9	557	5	

```

*   *DO YOU WANT THE TALLY OF DOCCUP SORTED
*   ((ALPHA)BETICALLY, OR ON FREQUENCY IN (ASCEND)ING
*   OR (DESCEND)ING ORDER -->? >descending
*   !TYPE ALPHA,ASCEND,OR DESCEND -->? >descend
*   *SHALL WE NORMALIZE -->? >no
*   *HOW MAN OF THE 13 ENTRIES DO YOU WANT PRINTED--->? >all
*   TALLY OF DOCCUP
*

```

DOCCUP	CUM %	% CUM	FREQ	FREQ	
10	35.3	35.3	200	200	+++++
07	55.7	20.5	316	116	+++++
00	74.30	18.5	421	105	+++++
03	82.7	8.5	469	48	+++++
08	88.5	5.8	502	33	+++++
11	92.4	3.9	524	22	+++++
06	94.7	2.3	537	13	XXXXXXXXXXXX
05	96.30	1.6	546	9	XXXXXXXXXX
12	97.4	1.1	552	6	XXXXXX
09	98.2	.9	557	5	XXXXX
01	98.9	.7	561	4	XXXX
02	99.60	.7	565	4	XXXX
04	100	.4	567	2	XX

Figure 5.34d. Here we see how the TALLY module handles large numbers when instructed not to normalize. In this form the small scale variations become more graphic and the large are easily identified.

## 5.35 TRIM

Often a data field is defined to allow a certain number of characters to accommodate a few lengthy entries even though the majority of the data items in that field are much shorter in length. The TRIM module reads through the entries in the data vectors named in response to

```
*TYPE LABEL(S) TO BE TRIMMED --->? >
```

and then gives a table presenting a count of character lengths in that vector.

If the longest entry does not have as many characters as the defined field the following message is printed

```
VECTOR      XYZ
  DEFINED LENGTH =      NNN
  CHARACTERS USED =      MMM
```

The user is then asked

```
*SHALL WE CHANGE THE POINTERS --->? >
```

We may respond by saying NO in which case the defined length of the field will remain as is. A YES redefines the field length to be equal to the maximum number of characters used or a number between 1 and the maximum used and redefines the length to be that specified number.

At the conclusion of this operation for each of the designated vectors, control is returned to the main Omnidata module with the pointers in the label table modified according to the above instructions. Hopefully, this will achieve operational economy and space compression useful in such modules as DISPLAY, REPORT, and in the PRINT option in the SEARCH module.

In figure 5.35a, we see the dialogue with and the output from this module when it examined the entries in the author field (actually, multiple authors) of a bibliographic file. We see there that this field was defined to include 133 characters, and that we actually made use of 129 of them—though only in 2 cases. In this circumstance, there is nothing to be gained by resetting the pointers.

The table of frequencies does show, however, that all but 14 of the records contain only 65 characters. If we were dealing with data items in a parts list instead of authors of journal articles, it would not be difficult to abbreviate the information in 14 records and achieve a 50% reduction in the space allowed for this data item.

Information from this module is useful even when dealing with this data vector of authors. A little mental arithmetic on the frequencies would show that if we allotted only 39 characters to the author field, 50 out of the 84 would fit. We have seen bibliographic listings in which the space limitations were such as to preclude listing more than the first author and even then truncating some of them in a few instances.

```

* * * * *
*   *WHICH DATA BASE DO YOU WANT --->? >ftjpcrd
*
* GOOD MORNING, WELCOME TO OMNIDATA
* * *NOTE-OMNIDATA KEEPS A RECORD OF WHO USED WHICH* * *
*   MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
*
* FILE FTJPCRD CONTAINS 11 DATA ITEMS FOR 84 RECORDS.
*   *TYPE A MODULE ANME AND/ OR INSTRUCTIONS
*   --->? >trim
*
*   TYPE LABEL(S) TO BE TRIMMED --->? >authors
* VECTOR AUTHORS DEFINED AS 133 CHARACTERS
* LENGTH FREQUENCY
* 0 1
* 12 2 43 1
* 13 (A) 2 52 2
* 14 1 55 1
* 16 3 56 1
* 17 5 62 1
* 18 2 63 1
* 19 2 64 1
* 20 1 65 8
* 21 2 66 3
* 22 1 67 1
* 23 2 68 1
* 27 1 70 1
* 28 1 73 2
* 29 1 81 1
* 30 3 86 1
* 31 2 105 1
* 32 5 122 1
* 33 1 129 2
* 34 1
* 35 2
* 36 2
* 37 2
* 38 3
* 39 2
* 41 1
* 45 1
* 46 2
* * * * *

```

VECTOR AUTHORS  
DEFINED LENGTH = 133  
CHARACTERS USED = 129  
\*SHALL WE CHANGE THE POINTER --->? >no

Figure 5.35a. Here we see the dialogue with and output from the TRIM module. At [A] we see that the author field was blank in one record, and that 129 of the positions assigned to that data item are filled—albeit in only 2 records. See the text for a further discussion of the utility of this frequency distribution.

If we wish to play such games, Omnidata facilitates them even after the file has been defined. If we wished such a compact printout we have at least two alternatives. We could EXTRACT from the defined author field the first 30 characters, call it SAUTHOR, and display that field instead of the original author field. Alternatively, we can use the SHORT instructions in the REPORT module and achieve the same result.

This module simply resets pointers in the file; it does not rewrite the file to compress it. In order to achieve more efficient storage of the data, it is necessary to use the ABRIDGE module.

\* \* \* N O T E S \* \* \*

\* \* N O T E S \* \*

## 6. Descriptions of the Utility Modules

In this section we describe the modules necessary for efficient management of data files as distinct from modules that select information from the files or perform data analysis. Some of these modules will be of interest only to the originator or builder of the file or the person whose job it is to correct, update, or otherwise maintain the file. Others will be of interest to the user as well.

The following comprise the utility modules:

ANNEX	ATTACH	BLANKS	CHECKSUM
DICTIONARY	MOVE	SCREEN	USERS

A few words are in order here concerning the absence of modules for deleting and inserting records and for a merge module. These are not needed because modules already exist to perform these functions. The SEARCH module will *delete* records from a file if they satisfy the search criteria based on information in the file. If records are to be deleted on the basis of their position in the file rather than their content, use can be made of the MOVE module.

The insertion of new records in an existing file can be achieved in two ways. Both require the use of two modules. If the new records are in a file or can be placed on a file from tape or cards, the STACK module will append one of the files to the other. After this the files can be sorted to put the new records in their proper place. If it is more convenient to supply the new records from the keyboard, the ANNEX module will place the new records at the end of the designated file. A subsequent SORT operation will put them in their proper order.

In our view, the requirement for moving records from their current location in a file arises largely when errors are found in one or more of the sort keys after the file has been sorted. If these errors are corrected via UPDATE, the file is no longer in proper sequence. When many records have to be moved, the easiest way is to sort the file again. If the file is large and the number of records to be moved is small, it is inefficient to sort the file again since sorting is expensive. For this reason we have included a module called MOVE.

## 6.1 ANNEX

ANNEX permits the user to add records to an existing Omnidata file on-line. If the user has already performed some operations on his file using other modules, the file in core is a temporary scratch file (resident in either TEMP, RTEMP, or SCRATCH). In such instances, the ANNEX operation may proceed directly. If, however, the user calls ANNEX immediately after entering Omnidata, he is given the message

YOU PRESENTLY HAVE YOUR ORIGINAL FILE . DO YOU  
WISH TO ANNEX RECORDS TO IT, OR TO A COPY?

\*TYPE O(RIGINAL) OR C(OPY)--->? >

This is in keeping with the authors' belief that the original file should never be tampered with in Omnidata except in rare instances and then only on specific instructions from the user. A file manager who wants the ability to himself ANNEX his file directly but keep others from doing so may make use of a write key.

The program next asks

\*DO YOU WANT PROMPTING MODE--->?

If the response to this question is YES, the program supplies each label in turn and waits for the user to input the data. At the completion of all the pieces of information the user may decide to make corrections by answering YES to the question

\*ANY CORRECTIONS--->? >

If he does answer YES, the program responds

\*TYPE LABEL, INFORMATION --->? >

and continues to accept corrections until a response of END is encountered. At this time the record is entered in the file and the user is asked again

\*DO YOU WANT PROMPTING MODE --->? >

for the possible input of another record, either using prompting or not.

If the user has elected not to use the prompting capability, he is asked to

\*TYPE LABEL, INFORMATION --->? >

and he can enter one or more data items. He ends the operation with a response of END and control switches to the question about the prompting mode. Any data item which he does not input remains blank for that record in the file. Typing errors can be corrected by repeating a label, followed by the correct information.

In either the prompting mode or the non-prompting mode, certain diagnostics are given for inappropriately entered information. For example, if alphabetic data is supplied for a field which has been defined as numeric, the user is asked to reenter the information. Also, if the user's input exceeds the defined length of a field, that input is truncated and the user is shown exactly how much of the input was entered in the given vector. Here the user has the option of allowing the truncated information to stand or supplying a shortened version.

```

* * * * *
* FILE FTTRAVEL CONTAINS 66 DATA ITEMS FOR 306 RECORDS.*
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->?annex
*
* YOU PRESENTLY HAVE YOUR ORIGINAL FILE. DO YOU
* WISH TO ANNEX RECORDS TO IT, OR TO A COPY?
*   *TYPE O(RIGINAL) OR C(OPY)--->? >O
*   *DO YOU WANT PROMPTING MODE--->? >yes
*   *REC --->? >79158
*   *NAME --->? >bertocci ugo
*   *AGENCY --->? >nbs
*   *DIVSEC --->? >312.04
*   *FROMYR --->? >77
*   *FROMMO --->? >03
*   *FROMDA --->? >30
* 30 MUST BE NUMERIC — RETYPE
*   *FROMDA --->? >30
*   *TOMO --->? >05
*   *TODA --->? >10
*   *TRAVELCODE --->? >20.3
*   *UNIT1 --->? >3120042
*   *COST1 --->? >0
*   *UNIT2 --->? >0
*   *S1CITY --->? >sofia
*   *S1COUN --->? >bg
*   *S1POA --->? >n
*   *S1OUA --->? >n
*   *S1PURPOSE --->? >to visit the central inst. of,&
* ? >electrochem power and inst. of phys. chem. of,&
* ? >bulgaria
*   *S2FYR --->? >end
*   *ANY CORRECTIONS --->? >no
*   *DO YOU WANT PROMPTING MODE --->? >yes
* * * * *

```



Figure 6.1a. Here we see the initiation of an operation to add new records directly to an existing file, FTTRAVEL, via the ANNEX module. Note at [A] how this module alerts the user when an alphabetic character is entered in a strictly numeric field. The reverse error—a number in place of a letter—is not caught because numerals are allowed in alphanumeric fields.

In a well-designed data base this problem should arise, if at all, only in data fields containing textual material. There the input can often be shortened by judicious editing or by using abbreviations or contractions.

When the user types END in response to the question

\*DO YOU WANT PROMPTING MODE --->? >

the module tells the user how many records have been added and how to get a listing of them as follows:

YOU HAVE ADDED RECORDS XXX-YYY  
TO LIST, CALL DISPLAY AND SKIP TO XXX.

Control is then switched back to Omnidata.

Please recall that unless the original file was present when ANNEX was called, a temporary file is in core and the SAVE module must be called to permanently catalog the file with the annexed records.

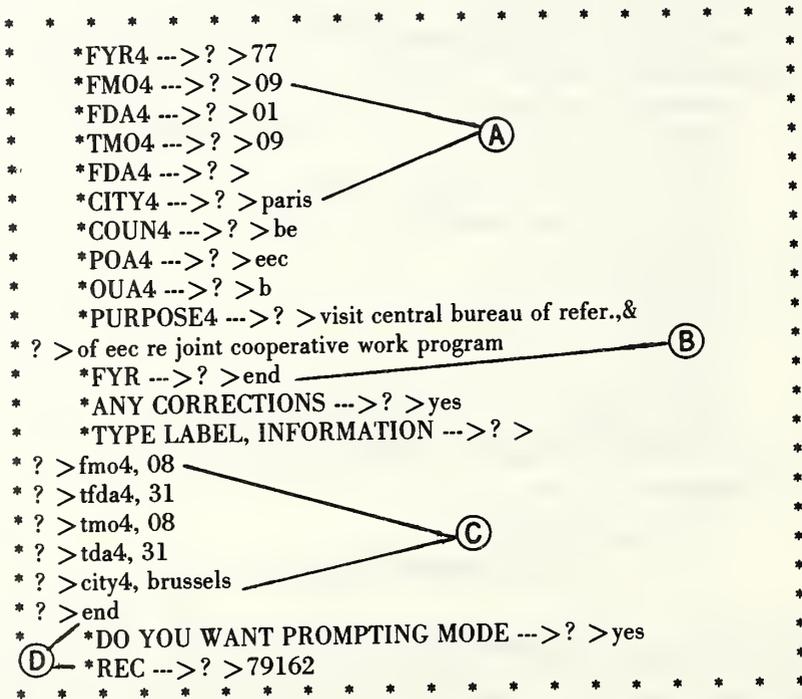


Figure 6.1b. Here we see how the five input mistakes at [A] are corrected at [C] after the input for this record is ended at [B]. When the corrections are terminated at [D], the system is ready for the next record.

In the figures that follow we see how records are entered into a travel file which can accommodate detailed information on as many as five cities per trip. In figure 6.1a we see data entered for the main portion of a record and for the data associated with the first stop on the itinerary. The request for input at [B] starts the entry for data relevant to the second stop. As there is no second stop on this trip, the input is terminated by typing END.

In figures 6.1b and 6.1c, we see how this module allows the user to correct the input for typing errors or when the length of the input exceeds the allotted space.

```

* * * * *
*   *COUN1 --->? >de
*   *POA1 --->? >icru
*   *OUA1 --->? >n
*   *PURPOSE1 --->? >attend meetings of icru report,&
* ? >com. of microdosimetry, and 3rd intl.,&
* ? >symposium on neutron dosimetry
* NOTE: TRUNCATED TO ATTEND MEETINGS OF ICRU REPORT COM. *
* OF MICRODOSIMETRY AND 3RD INTL. SYMPOSIUM ON NEUTRON
* DOSI.
*   *FYR2 --->? >end
*   *ANY CORRECTIONS --->? >yes
*   *TYPE LABEL, INFORMATION --->
* ? >purpose1, attend mtg. of icru report com. of,&
* ? >microdoseimetry and 3rd intl. symp. on neutron dosi
* ? >end
*   *DO YOU WANT PROMPTING MODE --->? >end
*   *YOU HAVE ADDED RECORDS 307-323
*   *TO LIST, CALL DISPLAY AND SKIP TO 307
*   *DON'T FORGET TO CALL SAVE
* * * * *

```

Figure 6.1c. Here we see how the system alerts the user at [A] that his input has been truncated, how judicious abbreviations at [B] solve the problem, and how the update operation is terminated at [C]. At [D] the module tells the user how to get a listing of the new records and at [E] we have a reminder to save the file. The reminder at [E] is not printed when records are annexed to the original file.

## 6.2 ATTACH

This module concatenates the corresponding records from two Omnidata files having a common record identifier. An interesting application for this module is the generation of a training incidence file where it is desirable to save look-up, keyboarding, and subsequent proofing by pulling the desired information from the main personnel file and combining it with the data for the specific training action.

If we had training actions for 100 persons punched up in file A and the social security (SS) number was used to identify the records, we would first SEARCH the main file for the list of 100 numbers. Next we would ABRIDGE the file to contain the SS number and the pertinent data items we wish to transfer to the training file. At the conclusion of the abridge operation we would have a file B with 100 records corresponding to those in file A.

The ATTACH module can combine these two files by concatenating the records having the same SS number if we tell it that the SS number is the key. Since files A and B are both in Omnidata form each has its own labels and pointers. The labels and their pointers from the first file will be carried over to the composite file. These will be augmented by labels from the second file with an adjustment of pointers to show their location in the longer composite record.

In figure 6.2a we see the dialogue for a successful operation to ATTACH the files FTR and FPERS100. This operation succeeded because both files were in proper sequence and the identification fields matched record for record.

In figure 6.2b we see how this module copes with discrepancies in the files. In the first instance if the number of records does not agree, the user has the option to terminate the operation at [A] by typing NO or STOP or of continuing by typing YES. If the choice is to continue, the module will check the files to see if they contain the required labels. If they do not, the operation is stopped with the following comment.

**\*FILE XYZ DOES NOT CONTAIN THE LABEL ABC**

If the labels agree, the operation progresses until there is a discrepancy in the identifier fields in corresponding records. At that point the operation is halted, the offending record number is printed, and the user has the option of saving the partial composite file.

In order to resolve the discrepancy, it will be necessary for the user to DISPLAY the offending records. What action is taken subsequently depends on the nature of the discrepancy in the files.

```

* * * * *
* FILE FTR CONTAINS 25 DATA ITEMS FOR 100 RECORDS
* --->? > attach
* *TYPE THE NAME OF THE FILE TO BE ATTACHED
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? > fpers100
* *TYPE LABELS FOR THE COMMON FIELD
* --->? > ss#,ssnum
* *TYPE NEW FILE NAME --->? > ftrain100
* *THE SAVED FILE FTRAIN100 CONTAINS 30 DATA ITEMS
* ?FOR 100 RECORDS
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* * * * *

```

(A)

Figure 6.2a. Here we see the dialogue for a successful operation to attach two consonant files. The labels at [A] are those in files FTR and FPERS100 respectively. The combined file will carry the label SS#.

```

* * * * *
* FILE FTR CONTAINS 25 DATA ITEMS FOR 100 RECORDS
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? > attach
* *TYPE THE NAME OF THE FILE TO BE ATTACHED
* --->? > fpab
* *FILE FPAB CONTAINS 98 RECORDS
* SHALL WE PROCEED --->? > yes
* *TYPE LABELS FOR THE COMMON FIELD
* --->? > ss#,ssnum
* *TYPE NEW FILE NAME --->? > ftrain
* *NOTE SS#8125793 DOES NOT AGREE WITH
* *SSNUM 8125739 IN RECORD 75
* *SHALL WE SAVE THE PARTIAL FILE? --->? > yes
* *THE SAVED FILE TTRAIN CONTAINS 30 ITEMS FOR 74 RECORDS
* *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? > display
* * * * *

```

(A)

(B)

Figure 6.2b. Here we see at [A] how the ATTACH module responds when the number of records in the two files does not agree. If the labels supplied at [B] are not proper, this module types out an appropriate message and terminates the operation. The subsequent dialogue shows how the module handles a discrepancy in the file and identifies the offending records.

### 6.3 BLANKS

When the user requests the module BLANKS, one pass through the data file is made and the number of totally blank entries is tallied for each defined label. A report is printed, giving only the labels where blank entries occur, and the number of such vectors. If no field has any blanks, the user receives the message that 'NO TOTALLY BLANK FIELDS OCCUR IN ANY VECTOR' Figure 3a shows a sample run of this module.

```
* * * * *
*
*   FILE FTBNM CONTAINS 34 DATA ITEMS FOR 1252 RECORDS.
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   --->? >blanks
*
*   LABEL                                ENTRIES
*   -----
*   2   TITLE                            3
*   3   JOURNAL                           4
*   6   A2                                669
*   7   A3                                1002
*   8   A4                                1151
*   9   A5                                1221
*   10  A6                                1242
*   11  A7                                1234
*   13  A5-7                              1209
*   18  TM2                               629
*   19  T1                                67
*
*   31  ATNUM1                            1043
*   32  ATNUM2                            1174
*   33  AT                                1136
*   34  LAB                               305
*
*   CPU SEC IN BLANKS = 82.6252
*   CPU SEC = 84.7024 TIM = 12:10:33
*
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
*   --->? >stop
*   PROGRAM STOPPED.
* * * * *
```

Figure 6.3a. Here we see how the BLANKS module reports the number of missing entries in certain of the data fields in this annotated bibliography. Since items 1, 4, 5, 12, and 14-17 are missing from this list, we can assume that all records in the file are complete in regard to those data fields.

## 6.4 CHECKSUM

Where file integrity is an important consideration, Omnidata provides a tool to determine if each record of the file is in the same condition as it was when generated or last validated. This is achieved by adding up the field data equivalent of the odd characters in each record and of the even characters in the record. These two sums are stored at the end of each of the data records.

When an Omnidata file is generated via the DEFINE module, the file manager is asked whether checksums should be generated. If the response is YES, the appropriate checksums are computed and appended to the end of each record. In addition, a checksum switch is set in the header portion of the file. Whenever the file is updated that switch alerts the UPDATE module to recompute and update the checksums for each of the changed records.

For files containing checksums, the file manager will need to keep a printed list of the checksums keyed to a unique record identifier. Such a list can be prepared via the REPORT module shortly after the data file is generated. New checksums resulting from updated records should be entered in this list by hand until such time as a new complete listing becomes necessary.

If checksums were always computed at the time the file is defined, there would be no need for a CHECKSUM module. This module is needed only when it is necessary to generate checksums for an existing Omnidata file which does not contain them. As the calculation of checksums is a relatively lengthy operation it should be performed in batch mode rather than on-line. If circumstances dictate performing this operation on-line, we suggest turning on the MONITOR.

When the checksums are computed, either during the DEFINE operation or via the module CHECKSUM, each logical record carries three pieces of information in addition to the two checksum numbers. These are: the date, the time of day, and the first six characters of the users RUN ID. The last of these identifies who performed the checksum operation. The above three pieces of information play an even more important role when the file is updated via the UPDATE module as they provide an important audit trail — telling when and by whom the record was updated.

## 6.5 DICTIONARY

This module adds to an Omnidata file of information required by the DESCRIBE and the SURVEY modules to describe the content and format of the data vectors associated with a particular data base in addition to giving a brief description of the file as a whole. When the data file contains encoded information this module will build a dictionary of codes for any or all of the data elements in the file. In the figure which follows we see how this module is used to supply information for generating a dictionary for a personnel file. Unless instructed to skip over some of the data items, this module asks for information for data elements in turn in the order in which they were defined.

```

* * * * *
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? > dictionary
*   *TYPE COMMENT ON FILE OR N(ONE) --->
* ?>THIS IS A SAMPLE OF A PERSONNEL,&
* ?>FILE CONTAINING 75 RECORDS
*
* 1. DIV
*   COMMENT --->? > none
*   CODE --->? > none
* 2. SS#
*   COMMENT --->? > social security number
*   CODE --->? > n
* 3. SEX
*   COMMENT --->? > n
*   CODE --->? > 1,male
*               > 2,female
* 4. CIT
*   COMMENT --->? > citizenship
*   CODE --->? > 1,u.s..
*               > 2,territorial
*               > 3,foreign
* 5. NAME
*   COMMENT --->? > 'last,first,mi,title'
*   CODE --->? > n
* 6. DOB
*   COMMENT --->? > date of birth
*   CODE --->? > n
* 7. YOB
*   COMMENT --->? > year of birth,&
*               > (20 for 1920 etc.)
*   CODE --->? > n
* * * * *

```

Figure 6.5a. Here we see how the DICTIONARY module asks for information to build up a dictionary file for use by the DESCRIBE and SURVEY modules. If information is not at hand for a group of labels, a response of SKIP TO 25 will cause this module to skip over the intervening labels.

## 6.6 MOVE

This module allows for moving data records in an Omnidata file. The need for this facility arises in two main instances: (1) if one wishes to order a file in a way that is not possible using the sorting sequence and (2) when typographical errors are found in the keys upon which a file had been sorted and the correction of these errors puts the file out of sort. If the file is large and the errors are few, it is more efficient to MOVE the offending records than it is to SORT the entire file again.

Instructions to the MOVE module are given in terms of record numbers in the original file. If the user supplies three record numbers (*a*, *b*, *c*) this module will move records *a* through *b* to follow record *c*. For each record or block of records to be moved the user must supply three numbers. The user may supply the move instructions in any order since this module sorts the instructions in order to rewrite the file from top to bottom.

If the user does not already know the record numbers associated with the data in the file on which this module is to operate, use can be made of the SEQUENCE module to provide this information in a systematic manner. When the file is large and the number of records to be moved is small, it may be possible to locate the required record number more easily by using the SKIP TO and BACK TO features of the DISPLAY module shown in figures 3.2c and 4b.

As the MOVE module must rewrite the entire file to accomplish its objective, it can and indeed does allow for duplicating records and even deleting records. A record or a block of records can be deleted simply by omitting the third number in the instruction triplet.

Figure 6.6a shows how information is supplied to this module in order to *move*, *duplicate*, and *delete* records from a file. Since all of the instructions are given in terms of the positions of the records in the original file, the user need not be concerned with any complications resulting from these moves, nor is it necessary to supply the instructions in any specified order. In the figure below we see in instructions [B], [C], and [D] how we are able to replace record 125 by record 20 and then move the modified block (120–135) to follow record 500. This operation will be performed properly regardless of the order in which the instructions are supplied.

```

* * * * *
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* ---> ? >move
*   *IF YOU TYPE A,B,C
*     RECORDS A THROUGH B WILL BE MOVED AFTER RECORD C
* ---> TYPE A,B,C
*   >12,16
* ? >25,25,10
* ? >87,87,12
* ? >125,125
* ? >20,20,125
* ? >120,135,500
*
* ? >end
*   *THIS FILE NOW CONTAINS XYZ RECORDS
*   CALL SAVE TO SAVE
*   *TYPE A MODULE NAME AND/OR INSTRUCTIONS
* * * * *

```

The diagram shows four annotations labeled A, B, C, and D, each enclosed in a circle. Lines connect these annotations to specific parts of the command syntax in the text above:

- Annotation **A** points to the second number '16' in the command `>12,16`.
- Annotation **B** points to the second number '125' in the command `>125,125`.
- Annotation **C** points to the second number '125' in the command `>20,20,125`.
- Annotation **D** points to the second number '500' in the command `>120,135,500`.

Figure 6.6a. Here we see the dialogue with the MOVE module where at [A] the absence of the third number causes records 12 to 16 to be deleted from the file. A careful examination of the above instructions will reveal that the module allows nesting of instructions .

## 6.7 SCREEN

The SCREEN module examines a designated data vector in each record in the file and reports the number of occurrences of each unique character in each character position specified. The output is given in the form of a two-dimensional table with the characters listed in rows, sorted on their field data numeric equivalents and with each column representing a different position in the data vector. The last column gives the total for the characters in all the columns screened thus far.

The user provides the name of the data vector to be screened in response to the question:

\*TYPE LABEL TO BE SCREENED, OR END --->? >

Assuming, then, that an appropriate label is input, the user is asked to

\*TYPE STARTING CHARACTER, NUMBER OF CHARACTERS  
TO SCREEN, OR END --->? >

Needless to say, the starting character must be a number less than or equal to the length of the vector. Also, a maximum of 10 characters can be screened in one pass. Diagnostics are given if the user has input a response to the above question which is impossible to fulfill.

After the designated columns are screened, the above question is repeated giving the user the opportunity to screen additional character positions in the same data vector. When the input to the question is END, indicating that no more screening for that vector is desired, the user is asked

\*DO YOU WANT A HISTOGRAM OF THE TOTAL —  
YES OR NO --->? >

Following this, the question asking for a label to be screened is repeated giving the opportunity to perform the screening operation on a different data vector. If the response is END, control is returned to the supervisory module. Figures 6.7a et seq. illustrate the results of screening certain data fields in a crystal data file.

```

* * * * *
* GOOD AFTERNOON, WELCOME TO OMNIDATA
* *NOTE - OMNIDATA KEEPS A RECORD OF WHO USED WHICH
*   MODULE(S) ON WHICH FILE(S) AT WHAT TIME OF THE DAY
* FILE ANOR-ORG CONTAINS 18 DATA ITEMS FOR 668 RECORDS.
*
*   TYPE A MODULE NAME AND/OR INSTRUCTIONS
* --->? >labels,screen
* THE FILE ANOR-ORG CONTAINS DATA LABELED AS FOLLOWS:
* 4   A       7   ALPHA   5   B       8   BETA   6   C
* 11  DM      12  DX      14  FORMULA 9   GAMMA 2   I OR O
* 15  IDNUM   16  M OR N  18  NAME    3   R1     17  R2
* 13  SG      1   SYSTEM  10  Z
*   TYPE LABEL TO BE SCREENED, OR END --->? >r1
*   *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,
*   OR END --->? >1,8
*           DISTRIBUTION OF CHARACTERS IN R1
*
*                   ***POSITION***
* CHAR   1     2     3     4     5     6     7     8     TOTAL
* -----
* 0  668  0  0  0  0  0  0  668  1336
* 0  (A) 0  664 0  5  50  65  59  0  843
* 1  0  4  0  30  60  60  69  0  223
* 2  0  (B) 0  0  21  47  69  61  0  198
* 3  0  0  0  20  62  70  75  0  227
* 4  0  0  0  33  85  63  62  0  243
* 5  0  0  0  49  65  71  54  0  239
* 6  0  0  0  69  83  69  68  0  289
* 7  0  0  0  110 73  69  66  0  318
* 8  0  0  0  144 66  62  84  0  356
* 9  0  0  0  187 77  70  70  0  404
* (C) 0  0  0  668 0  0  0  0  668
*
* *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,
* OR END --->? >end
* * * * *

```

Figure 6.7a. In this use of the SCREEN module as the R1 data vector we see at [A] that the first position is blank in all records, at [B] that all but four records are blank in the second position and that the others contain ones. The information at [C] tells us that the decimal point is present in the third position in each record. Thus, we know that except for the four records noted above, the values of R1 are less than unity.

```

* * * * *
* LABEL TO BE SCREENED, OR END ---->? >formula
* *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,
* OR END --->? >1,12
* !MAXIMUM OF 10 CHARACTERS SCREENED AT ONCE- RETYPE LINE
* *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,
* OR END --->? >1,10
*
*
* DISTRIBUTION OF CHARACTERS IN FORMULA
*
* * * * POSITION * * *
* CHAR 1 2 3 4 5 6 7 8 9 10 TOTAL*
*-----*
* [ 0 32 3 1 34 1 1 0 1 0 73 *
* ] 0 0 0 0 0 0 0 0 0 1 1 *
* 666 547 99 72 205 380 91 179 166 289 2594*
* A 0 0 9 17 0 1 1 0 0 0 28 *
* B 0 0 12 10 0 4 1 1 2 5 35 *
* C 1 0 385 73 27 43 79 32 69 33 742 *
* I 0 0 0 15 2 0 0 2 0 0 19 *
* E 2 0 0 27 0 0 2 1 0 1 31 *
* F 0 0 5 0 0 2 2 1 1 0 11 *
* G 0 0 1 0 0 0 0 0 0 0 11 *
*
* Z 0 0 9 1 0 0 2 0 0 0 10 *
* ) 0 0 0 0 0 0 3 18 20 19 60 *
* - 0 0 0 0 0 0 1 1 0 1 3 *
* + 0 0 0 0 0 3 6 8 25 21 69 *
* = 0 0 0 0 0 0 0 0 1 1 2 *
* ( 1 83 19 7 80 17 19 16 11 37 290 *
* : 0 0 0 0 0 0 1 0 0 0 1 *
* 0 0 0 0 35 2 1 18 39 1 96 *
* 1 0 1 0 130 28 2 35 71 14 4 285 *
* 2 0 4 0 56 47 15 15 91 58 43 329 *
*
* 9 0 0 0 11 21 5 11 5 4 5 62 *
* . 0 0 0 0 1 0 0 0 0 1 2 *
*
* *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,
* OR END ---->? >11,8
*-----*

```

Figure 6.7b. In this illustration of the operation of the SCREEN module, we see at [A] how the user is reminded that at most 10 character positions can be screened in one pass through the file. At [B] we see that all but two records have a blank in the first position of the data field. If the statistics at [C] and [D] are thought to reflect errors in these two records, they can easily be isolated using the SEARCH module.

```

* * * * *
* *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,*
* OR END --->? >11,8
*
*
*          DISTRIBUTION OF CHARACTERS IN FORMULA
*          * * * POSITION * * *
* CHAR 11  12  13  14  15  16  17  18  TOTAL
*-----
* [      3   1   0   0   1   3   3   0   84
* ]      0   0   2   2  11   2   6   4   28
*      207 233 309 258 317 288 356 343 5005
* A      3   2   2   1   5   3   9   2   55
* B     10   1   1   2   1   6   4   0   60
* C     61  59  44  35  27  52  32  32 1024
* D      1   0   0   0   0   1   2   0   23
* E      2   2   2   1   3   2   2   0   43
*
* Z      0   0   0   1   0   0   1   1   13
* )     15  25  21  15  27  46  22  21  252
* -      0   0   1   6   1   0   1   0   12
* +     21  18  12  19  15  25  14  19  218
* =      0   0   1   0   2   0   0   1   6
* (      9  15   7  18   4   3  11   5  362
* :      1   2   2   1   1   1   2   0   11
* 0      2   1   1   4   1   3   2   1  111
* 1     10   7   7   4   8   1   4   2  328
* 2     45  67  43  55  57  48  47  31  722
* 3     14  23  13  18  20  18  12  10  304
* 4     24  26  15  14  19  11  12  18  306
* 5      9  15   3  12  10  12  11   5  191
* 6      7  22  21   7  12  14   7   8  249
* 7     11   3   3   5   1   1   0   1   77
* 8      8   2   1   5   7   4   2   8  119
* 9      1   4   0   1   1   1   0   0   70
*      0   0   0   0   5   1   1   2   11
*
* *TYPE STARTING CHARACTER, NUMBER OF CHARACTERS TO SCREEN,*
* OR END --->? >end
*
* * * * *

```

Figure 6.7c. Here we continue to screen the next 8 character positions. Note that the totals are for all of the 18 characters screened thus far.

## 6.8 USERS

Access to Omnidata files is controlled by requiring the data-base administrator to build into each file a list of accredited users and associated passwords. At the time an Omnidata file is generated, the DEFINE program calls the program USERS and allows the data-base administrator to supply the list of accredited users at that time.

It usually becomes necessary at a later time to add to or delete names from this list. This module has been written to achieve this important housekeeping operation. USERS operates as a stand-alone X BASIC program as well as an Omnidata module. In the figures which follow we illustrate the features of the independent X BASIC program. When called as a module in an Omnidata run, it operates in the same fashion, except that the first question at [A] is bypassed since the module already knows the name of the file on which it is to write.

```
* * * * *
* >old:users*
* READY
* >run
*
*   USERS          11:41:13    2 MAY 77
*
*   *WHICH DATA BASE DO YOU WANT --->? >fpers100
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >list
* NO USERS TO LIST
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >add
*   *TYPE NAME OR ACCOUNT NUMBER --->? >jh
*   *TYPE PASSWORD --->? >1112
*   *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
*   *TYPE NAME OR ACCOUNT NUMBER --->? >end
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >list
*
* NAME OR ACCOUNT NO.    PASSWORD    FLAG
* -----
* JH                      1112        0
*
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >stop
* PROGRAM STOPPED.
* * * * *
```

Figure 6.8a. Here we see how a user is accredited to a file via the program USERS. In actual operation the file name at [A] would be followed by read and write keys that are required to preserve the file security.

```

* * * * *
* >old:users*
*   READY
* >run
*
*   USERS           09:11:09       3 MAY 77
*
*   *WHICH DATA BASE DO YOU WANT --->? >fpers100
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >add
*   *TYPE NAME OR ACCOUNT NUMBER --->? >jh
*   *TYPE PASSWORD --->? >1112
*   *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
*   *TYPE NAME OR ACCOUNT NUMBER --->? >end
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >list
*
* NAME OR ACCOUNT NO.           PASSWORD   FLAG
* -----
*   JH                           JH           0
*   JH                           1112          0
*
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >delete
*   *TYPE NAME OR ACCOUNT NO. TO BE DELETED --->? >jh
*   *TYPE NAME OR ACCOUNT NO. TO BE DELETED --->? >end
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >list
*
* NAME OR ACCOUNT NO.           PASSWORD   FLAG
* -----
*   JH                           1112          0
*
*   TYPE ADD, DELETE, CHANGE, OR LIST --->? >stop
*   PROGRAM STOPPED.
* * * * *

```

Figure 6.8b. Here we see how to add accredited users to a file and how to delete them. Note that when duplicate names appear the delete operation removes the first one.

```

* * * * *
* >old:users*
*   READY
* >run
*
*   USERS          13:52:09      9 AUG 77
*
*   *WHICH DATA BASE DO YOU WANT --->? >ftjpcrd
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >list
*
*   NAME OR ACCOUNT NO.      PASSWORD      FLAG
*   -----
*   BJBM                      OK          0
*   A                          X          0
*   B                          Y          0
*   C                          Z          0
*   JH                         1112      1
*
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >change
*   *TYPE NAME OR ACCOUNT NO. FOR WHICH CHANGE DESIRED --->? >jh
*   *TYPE NAME OR ACCOUNT NUMBER --->? >jh
*   *TYPE PASSWORD --->? >1112
*   *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
*   *TYPE NAME OR ACCOUNT NO. FOR WHICH CHANGE DESIRED --->? >end
*   *TYPE ADD, DELETE, CHANGE,OR LIST --->? >stop
*   PROGRAM STOPPED.
*
*   TIME :    1.026
* * * * *

```

Figure 6.8c. Here we see how the USERS module is used to modify an entry in the list of accredited users.

```

* * * * *
* >old:users*
*   READY
* >run
*
*   *WHICH DATA BASE DO YOU WANT --->? >anor-inorg
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >add
*   *TYPE NAME OR ACCOUNT NUMBER --->? >b
*   *TYPE PASSWORD --->? >ok
*   *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >end
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >add
*   *TYPE NAME OR ACCOUNT NUMBER --->? >bj
*   *TYPE PASSWORD --->? >ok
*   *TYPE 1 FOR RESTRICTED, 0 FOR NORMAL --->? >0
*   *TYPE NAME OR ACCOUNT NUMBER --->? >end
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >list
*
*   NAME OR ACCOUNT NO.      PASSWORD      FLAG
*   -----
*   BJ                        OK              0
*
*   *TYPE ADD, DELETE, CHANGE, OR LIST --->? >end
*
*   TIME :      1.032
* * * * *

```

Figure 6.8d. Here we see at [B] how to end the input when an error is discovered before all of the data items are entered. Had we noticed the error at [A] after supplying the required answer at [B], the only way to correct it would be via the CHANGE option.

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO.  NBS Handbook 125	2. Gov't Accession No.	3. Recipient's Accession No.
4. TITLE AND SUBTITLE OMNIDATA: An Interactive System for Data Retrieval, Statistical and Graphical Analysis, and Data-Base Management -- A User's Manual		5. Publication Date September 1978 6. Performing Organization Code	
7. AUTHOR(S) Joseph Hilsenrath and Bettijoyce Breen		8. Performing Organ. Report No.	
9. PERFORMING ORGANIZATION NAME AND ADDRESS  NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, D.C. 20234		10. Project/Task/Work Unit No.	
12. Sponsoring Organization Name and Complete Address (Street, City, State, ZIP)  Same as item 9.		11. Contract/Grant No.	
13. Type of Report & Period Covered		14. Sponsoring Agency Code	
15. SUPPLEMENTARY NOTES  Library of Congress Catalog Card Number: 78-600076			
16. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.) The Omnidata system, consisting of 45 individual programs written in X BASIC, provides an interactive user-oriented facility for: data retrieval and report generation; plotting and other graphical analysis; arithmetic and statistical analysis; curve fitting and multiple linear regression; data coding and decoding; survey and questionnaire analysis; author, title, and keyword indexing of bibliographic files; a variety of univariate analyses and two-way crosstabulations; and numerous utility modules for file definition, file updating, and file maintenance. The SEARCH module which performs a serial search through a file allows for: the usual Boolean operations; string searching on text fragments, stems, or roots in either the anchored or unanchored mode; specification of syntactical order and proximity of words or phrases, as well as variable length <i>ellipsis</i> ; and ignoring one or more of a specified list of characters in its matching operation. Four of the modules interface with the OMNITAB II system for versatile plotting, very accurate least-squares fitting, and a comprehensive statistical analysis.			
17. KEY WORDS (six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons) Boolean search system; computer programs; curve fitting; data analysis; data base management; data retrieval; file handling; graphic analysis; IMS information retrieval; KWOC indexing; least-squares; linear regression; MIS; plotting; statistical analysis.			
18. AVAILABILITY <input checked="" type="checkbox"/> Unlimited  <input type="checkbox"/> For Official Distribution. Do Not Release to NTIS  <input checked="" type="checkbox"/> Order From Sup. of Docs., U.S. Government Printing Office Washington, D.C. 20402, Stock No. 003-003-01972-1  <input type="checkbox"/> Order From National Technical Information Service (NTIS) Springfield, Virginia 22151		19. SECURITY CLASS (THIS REPORT)  UNCLASSIFIED	21. NO. OF PAGES  294
		20. SECURITY CLASS (THIS PAGE)  UNCLASSIFIED	22. Price

